

---

# Documentazione di LVGL

## v9.420250816 (Ita)

*Release 9.4*

**LVGL Community**

(Traduzione: [Baldassarre Cesarano](#))

16 ago 2025

---

## Indice

---

<b>1</b>	<b>Introduzione</b>	<b>1</b>
<b>2</b>	<b>Per iniziare</b>	<b>7</b>
<b>3</b>	<b>Esempi</b>	<b>14</b>
<b>4</b>	<b>Integrazione e Driver</b>	<b>106</b>
<b>5</b>	<b>Funzionalità Comuni dei Widget</b>	<b>240</b>
<b>6</b>	<b>Tutti i Widget</b>	<b>319</b>
<b>7</b>	<b>Main Modules</b>	<b>448</b>
<b>8</b>	<b>XML - Declarative UI</b>	<b>531</b>
<b>9</b>	<b>Moduli Ausiliari</b>	<b>565</b>
<b>10</b>	<b>3rd-Party Libraries</b>	<b>618</b>
<b>11</b>	<b>Debugging</b>	<b>642</b>
<b>12</b>	<b>Contribuire</b>	<b>649</b>
<b>13</b>	<b>Change Log</b>	<b>658</b>
<b>14</b>	<b>API</b>	<b>684</b>

# CAPITOLO 1

---

## Introduzione

---

LVGL (Light and Versatile Graphics Library) è una libreria grafica C gratuita e open source che fornisce tutto il necessario per creare un'interfaccia utente grafica embedded con elementi grafici facili da usare, splendidi effetti visivi e un ridotto utilizzo di memoria.

LVGL fornisce il "motore GUI" che gestisce tutte le interazioni dell'utente finale con l'applicazione, tra cui non solo la visualizzazione del contenuto del display, l'accettazione dell'input dell'utente e la sua conversione in eventi "catturabili" dall'applicazione, ma anche oltre 30 widget integrati, oltre 100 proprietà di stile, scrolling [scorrimento], animazioni, rendering software avanzato, supporto GPU integrato da MCU a OpenGL e molto altro. La combinazione di questi elementi consente di sviluppare interfacce utente in modo efficiente senza dover reinventare la ruota.

L'utilizzo di LVGL in un progetto firmware riduce efficacemente il lavoro dello sviluppatore della GUI alla sua definizione e alla fornitura di hook di eventi appropriati per svolgere il prezioso lavoro della GUI.

LVGL può utilizzare GUI definite al 100% nel codice C scritto dall'utente (per cui qualsiasi aggiornamento alla GUI richiede la modifica del codice C e il re-flashing del firmware), oppure definizioni di GUI fornite in file XML (per cui gli aggiornamenti alla GUI richiedono solo la modifica dei file XML che definiscono la GUI per quel firmware), o qualsiasi combinazione di queste due opzioni adatta al progetto.

### 1.1 Caratteristiche Principali

- Elementi costitutivi potenti come *pulsanti, grafici, elenchi, slider, immagini*, ecc.
- Grafica avanzata con animazioni, anti-aliasing, opacità, scorrimento fluidog
- Vari dispositivi di input come touchpad, mouse, tastiera, encoder, ecc.
- Supporto multilingua con codifica UTF-8
- Supporto multi-display, anche con formati di colore misti
- Elementi grafici completamente personalizzabili con stili simili a CSS
- Indipendente dall'hardware: utilizzabile con qualsiasi microcontrollore o display
- Scalabile: in grado di funzionare con poca memoria (64 kB Flash, 16 kB RAM)
- *OS*, memoria esterna e *GPU* sono supportati ma non obbligatori
- Funzionamento con buffer di frame singolo anche con effetti grafici avanzati
- Scritto in C per la massima compatibilità (compatibile con C++)
- *Simulatore* per avviare la progettazione di GUI embedded su un PC senza hardware integrato

- Il codice utente sviluppato tramite simulatore può essere condiviso con il firmware per rendere più efficiente lo sviluppo dell'interfaccia utente.
- Binding a [MicroPython](#)
- Tutorial, esempi, temi per la progettazione rapida di interfacce grafiche (GUI)
- Documentazione disponibile online
- Gratuito e open source con licenza MIT
- Gratuito per progetti commerciali

### 1.1.1 Requisiti

In pratica, qualsiasi controller moderno in grado di pilotare un display è adatto all'esecuzione di LVGL. I requisiti minimi sono:

- Microcontrollore o processore a 16, 32 o 64 bit
- Velocità di clock consigliata >16 MHz
- Flash/ROM: >64 kB per i componenti essenziali (>180 kB consigliati)
- RAM:
  - Utilizzo di RAM statica: ~2 kB a seconda delle funzionalità utilizzate e dei tipi di Widget
  - stack: > 2kB (> 8 kB consigliati)
  - Dati dinamici (heap): >2 KB (>48 kB consigliati se si utilizzano molti widget GUI). Impostato da [LV\\_MEM\\_SIZE](#) in `lv_conf.h`.
  - Buffer di visualizzazione: > "Risoluzione orizzontale" pixel (>10 X "Risoluzione orizzontale")
  - Un frame buffer nell'MCU o in un controller video esterno
- Compilatore C99 o successivo
- Conoscenza base del C (o C++):
  - puntatori.
  - struct.
  - callback.

#### Nota

*L'utilizzo della memoria può variare a seconda dell'architettura, del compilatore e delle opzioni di build.*

### 1.1.2 Licenza

Il progetto LVGL (inclusi tutti i repository) è concesso con la [licenza MIT](#). Questo significa che lo si può utilizzare anche in progetti commerciali.

Non è obbligatorio, ma apprezzeremmo molto se si scrivesse qualche parola sul proprio progetto nella categoria [My projects](#) del forum o si inviasse un messaggio privato a [lvgl.io](#).

Sebbene LVGL sia disponibile gratuitamente, dietro c'è un enorme lavoro. È stato creato da un gruppo di volontari che lo hanno reso disponibile a tutti nel loro tempo libero.

Per rendere sostenibile il progetto LVGL, preghiamo di considerare la possibilità di [Contribuire](#). Si può scegliere tra diversi modi per contribuire. Consultare [Contribuire](#), ad esempio scrivendo semplicemente un tweet su di sé usando LVGL, correggendo bug, traducendo la documentazione o addirittura diventando un manutentore.

### 1.1.3 FAQ

#### Dove posso porre domande?

Si possono porre domande nel forum: <https://forum.lvgl.io/>.

Utilizziamo [GitHub issues](#) per le discussioni relative allo sviluppo. Si dovrebbero utilizzarli solo se la domanda o il problema è strettamente correlato allo sviluppo della libreria.

Prima di postare una domanda, leggere questa sezione FAQ, poiché si potrebbe trovare la risposta al problema anche qui.

#### Questo MCU/hardware è supportato?

Tutti i MCU in grado di pilotare un display tramite porta parallela, SPI, interfaccia RGB o altro e che soddisfano i [Requisiti](#) sono supportati da LVGL.

Questo include:

- MCU "comuni" come STM32F, STM32H, NXP Kinetis, LPC, iMX, dsPIC33, PIC32, SWM341 ecc.
- Moduli Bluetooth, GSM e Wi-Fi come Nordic NRF, Espressif ESP32 e Raspberry Pi Pico W
- Linux con device frame buffer come /dev/fb0. Questo include computer Single-board come il Raspberry Pi
- Qualsiasi altro dispositivo con una MCU sufficientemente potente e una periferica per pilotare un display

#### Questo display è supportato?

LVGL richiede solo una semplice funzione driver per copiare un array di pixel in una determinata area del display. Se lo si può fare col proprio display, lo si può usare con LVGL.

Alcuni esempi dei tipi di display supportati:

- TFT con profondità di colore a 16 o 24 bit
- Monitor con porta HDMI
- Piccoli display monocromatici
- Display in scala di grigi
- anche matrici di LED
- o qualsiasi altro display in cui sia possibile controllare il colore/stato dei pixel

Consultare la sezione [Display \(lv\\_display\)](#) per ulteriori informazioni.

#### LVGL non si avvia, si blocca casualmente o non viene visualizzato nulla sul display. Quale potrebbe essere il problema?

- Provare ad aumentare [LV\\_MEM\\_SIZE](#).
- Assicurarsi che il display funzioni senza LVGL. Ad esempio, coloralo in rosso all'avvio.
- Abilitare [Logging](#).
- Abilitare le asserzioni in `lv_conf.h` (`LV_USE_ASSERT_...`).
- Se si usa un RTOS:
  - Aumentare la dimensione dello stack del task che chiama `lv_timer_handler()`.
  - Utilizzare uno dei metodi per la gestione dei thread descritti in [Considerazioni sui Thread](#).

## Il driver video non viene chiamato. Cos'è sfuggito?

Chiamare `lv_tick_inc(x)` come prescritto in [Tick Interface](#) e chiamare `lv_timer_handler()` come prescritto in [Gestione del Timer](#).

Per saperne di più, consultare le sezioni [Tick Interface](#) e [Gestione del Timer](#).

## Perché il driver video viene chiamato una sola volta? Viene aggiornata solo la parte superiore del display.

Chiamare `lv_display_flush_ready(drv)` alla fine del proprio "display flush callback" come indicato nella sezione [Flush Callback](#).

## Perché si vede solo spazzatura sullo schermo?

Probabilmente c'è un bug nel driver del display. Provare il seguente codice senza usare LVGL. Si dovrebbe vedere un quadrato con una sfumatura rosso-blu.

```
#define BUF_WIDTH 255
uint16_t buf[BUF_WIDTH];
uint32_t i;
for(i = 0; i < BUF_WIDTH; i++) {
    lv_color_t c = lv_color_mix(lv_color_hex(0xffff0000), lv_color_hex(0x00ffff), i);
    buf[i] = lv_color_to_u16(c);

    lv_area_t a;
    a.x1 = 5;
    a.x2 = a.x1 + BUF_WIDTH - 1;
    a.y1 = 10 + i;
    a.y2 = 10 + i;
    my_flush_cb(NULL, &a, (void*) buf);
}
```

## Perché si vedono colori senza senso sullo schermo?

Il formato colore LVGL configurato probabilmente non è compatibile con il formato colore del display. Controllare `LV_COLOR_DEPTH` in `lv_conf.h`.

## Come si può velocizzare l'interfaccia utente?

- Attivare l'ottimizzazione del compilatore e abilitare la memorizzazione nella cache di istruzioni e dati, se disponibile sull'MCU.
- Aumentare la dimensione del buffer di visualizzazione.
- Utilizzare due buffer di visualizzazione e svuotare il buffer con DMA (o una periferica simile) in background.
- Aumentare la velocità di clock della porta SPI o parallela se si utilizzano per pilotare il display.
- Se il display ha una porta SPI, valutare la possibilità di passare a un modello con interfaccia parallela, poiché offre una velocità di elaborazione molto più elevata.
- Mantenere il buffer di visualizzazione nella RAM interna (non nella SRAM esterna) perché LVGL lo utilizza molto e dovrebbe avere tempi di accesso rapidi.

## Come posso ridurre l'utilizzo di flash/ROM?

Si possono disabilitare le funzionalità inutilizzate (come animazioni, file system, GPU, ecc.) e i tipi di widget in `lv_conf.h`.

Se si utilizza GCC/CLANG, è possibile aggiungere i flag di compilazione `-fdata-sections -ffunction-sections` e il flag del linker `--gc-sections` per rimuovere funzioni e variabili non utilizzate dal binario finale. Se possibile, aggiungere il flag del compilatore `-flto` per abilitare il link-time-optimisation insieme a `-Os` per GCC o `-Oz` per CLANG e le versioni più recenti di GCC.

## Come ridurre l'utilizzo di RAM?

- Ridurre la dimensione del *Display buffer*.
- Ridurre *LV\_MEM\_SIZE* in *lv\_conf.h*. Questa memoria viene utilizzata quando si creano widget come pulsanti, etichette, ecc.
- Per lavorare con *LV\_MEM\_SIZE* inferiori, è possibile creare widget solo quando necessario ed eliminarli quando non sono più necessari.

## Come si utilizza LVGL con un sistema operativo?

Per lavorare con un sistema operativo in cui le attività possono interrompersi a vicenda (in modo preventivo), è necessario assicurarsi che nessuna chiamata di funzione LVGL venga chiamata mentre è in corso un'altra chiamata LVGL. Esistono diversi modi per farlo. Consultare la sezione *Considerazioni sui Thread* per saperne di più.

### 1.1.4 Il Repository LVGL

#### Layout del repository

Tutti i repository del progetto LVGL sono ospitati su [GitHub](#).

Lì si trovano i seguenti repository:

- **lvgl**: La libreria stessa con numerosi [esempi](#) e [demo](#).
- **blog**: Sorgente del sito del blog
- **sim**: Sorgente del sito del simulatore online
- **lv\_port\_\***: Port LVGL per schede o ambienti di sviluppo
- **lv\_binding\_\***: Binding ad altri linguaggi

#### Policy di rilascio

I repository principali seguono le regole del [Semantic Versioning](#):

- Versione major: modifiche API incompatibili. Es. v5.0.0, v6.0.0
- Versione minor: funzionalità nuove ma retrocompatibili. Es. v6.1.0, v6.2.0
- Versione patch: correzioni di bug retrocompatibili. Es. v6.1.1, v6.1.2

Tag come vX.Y.Z vengono creati per ogni release.

#### Ciclo di rilascio

- Correzioni di bug: rilasciate su richiesta, anche settimanalmente
- Release minor: ogni 3-4 mesi
- Release major: circa una volta all'anno

#### Branch

I repository principali hanno almeno i seguenti branch:

- **master**: versione più recente, le patch vengono unite direttamente qui
- **release/vX.Y**: versioni stabili delle release minor

#### Changelog

Le modifiche sono loggiate in [Change Log](#).

## Versioni supportate

Prima della v8, l'ultima release minor di ogni serie principale era supportata per 1 anno. A partire dalla v8, ogni release minor è supportata per 1 anno.

Versione	Data di rilascio	Fine del supporto	Attivo
v5.3	1 Feb, 2019	1 Feb, 2020	No
v6.1	26 Nov, 2019	26 Nov, 2020	No
v7.11	16 Mar, 2021	16 Mar, 2022	No
v8.0	1 Jun, 2021	1 Jun, 2022	No
v8.1	10 Nov, 2021	10 Nov, 2022	No
v8.2	31 Gen, 2022	31 Gen, 2023	No
v8.3	6 Lug, 2022	1 Gen, 2025	No
v8.4	19 Mar, 2024	20 Mar, 2025	No
v9.0	22 Gen, 2024	22 Gen, 2025	No
v9.1	20 Mar, 2024	20 Mar, 2025	No
v9.2	26 Ago, 2024	26 Ago, 2025	Yes
v9.3	3 giugno 2025	3 giugno 2026	Yes

# CAPITOLO 2

---

Per iniziare

---

## 2.1 Imparare le Basi

### 2.1.1 Panoramica

LVGL (Light and Versatile Graphics Library) è una libreria grafica gratuita e open source che fornisce tutto il necessario per creare un'interfaccia utente grafica embedded con elementi grafici facili da usare, simili a quelli dei telefoni cellulari, splendidi effetti visivi e un ridotto utilizzo della memoria.

LVGL può essere considerato come una raccolta di file C e H inseribili in qualsiasi progetto per aggiungere funzionalità di interfaccia utente al prodotto.

Con l'aiuto di funzioni API coerenti e facili da apprendere, è possibile creare widget (pulsanti, slider, grafici, ecc.), personalizzarli, aggiungere eventi, layout o animazioni.

In base a queste impostazioni, LVGL renderizzerà un'immagine (utilizzando il suo motore di rendering software nativo o una GPU) e chiamerà una funzione di callback per visualizzare l'immagine renderizzata sul display. Questa funzione di callback è l'interfaccia principale tra LVGL e il display. La maggior parte del lavoro relativo al porting si concentra sulla scrittura di tale callback in modo efficace.

Questo capitolo illustrerà le basi per dare un'idea del funzionamento di LVGL e di come può essere utilizzato. Per maggiori dettagli su ciascuna funzionalità, visitare la pagina di documentazione dedicata.

### Data Flow di Base

1. **Inizializzazione del Driver:** È responsabilità dell'utente configurare l'orologio, i timer, le periferiche, ecc.
2. **Chiamata a `lv_init()`:** Inizializza LVGL stesso
3. **Creazione di dispositivi di visualizzazione e di input:** Si creano display (`lv_display_t`) e dispositivi di input (`lv_indev_t`) e si impostano i relativi callback
4. **Creazione dell'Interfaccia Utente:** Si chiamano le funzioni LVGL per creare schermate, widget, stili, animazioni, eventi, ecc.
5. **Chiamata a `lv_timer_handler()` in un loop:** Gestisce tutte le attività relative a LVGL:
  - aggiornare i(l) display,
  - leggere i dispositivi di input,
  - generare eventi in base all'input dell'utente (e altre cose),

- eseguire eventuali animazioni ed
- eseguire i timer creati dall'utente.

### Esempio di integrazione

Questo è solo un breve esempio di come aggiungere LVGL a un nuovo progetto. Per maggiori dettagli, consultare [Aggiungere LVGL al Proprio Progetto](#).

```
void main(void)
{
    your_driver_init();

    lv_init();

    lv_tick_set_cb(my_get_millis);

    lv_display_t * display = lv_display_create(320, 240);

    /*LVGL will render to this 1/10 screen sized buffer for 2 bytes/pixel*/
    static uint8_t buf[320 * 240 / 10 * 2];
    lv_display_set_buffers(display, buf, NULL, LV_DISPLAY_RENDER_MODE_PARTIAL);

    /*This callback will display the rendered image*/
    lv_display_set_flush_cb(display, my_flush_cb);

    /*Create widgets*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello LVGL!");

    /*Make LVGL periodically execute its tasks*/
    while(1) {
        lv_timer_handler();
        my_sleep(5); /*Wait 5 milliseconds before processing LVGL timer again*/
    }
}

/*Return the elapsed milliseconds since startup.
 *It needs to be implemented by the user*/
uint32_t my_get_millis(void)
{
    return my_tick_ms;
}

/*Copy the rendered image to the screen.
 *It needs to be implemented by the user*/
void my_flush_cb(lv_display_t * disp, const lv_area_t * area, uint8_t * px_buf)
{
    /*Show the rendered image on the display*/
    my_display_update(area, px_buf);

    /*Indicate that the buffer is available.
     *If DMA were used, call in the DMA complete interrupt*/
    lv_display_flush_ready();
}
```

## 2.1.2 Display

*Display* si riferisce all'hardware effettivo. Per collegare LVGL all'hardware, è necessario creare e inizializzare un oggetto `lv_display_t`.

LVGL supporta molti *driver integrati*, ma è facile anche inizializzare un display da zero (come mostrato sopra).

LVGL gestisce anche più display contemporaneamente.

## 2.1.3 Screen

Uno *Screen* è un widget LVGL creato su un *Display*. È un contenitore logico per altri widget. Un display può avere più screen, ma ce n'è sempre uno attivo, che può essere recuperato tramite `lv_screen_active()`. Restituisce un puntatore `lv_obj_t *`. Consultare [Lo Screen Attivo](#) per ulteriori informazioni.

Il modo più comune per creare uno screen è creare un *Widget di Base* con un genitore `NULL`. Ad esempio,

```
lv_obj_t * my_screen = lv_obj_create(NULL);
```

Uno screen può essere caricato in questo modo: `lv_screen_load(my_screen)`

## 2.1.4 I Widget

I widget sono gli elementi costitutivi fondamentali dell'interfaccia utente. Per esempio: *Button* (`lv_button`), *Slider* (`lv_slider`), *Drop-Down List* (`lv_dropdown`), *Chart* (`lv_chart`), ecc.

I widget possono essere creati dinamicamente chiamando le rispettive funzioni "create". La funzione `create` restituisce un puntatore `lv_obj_t *` utilizzabile per configurare il widget in un secondo momento.

Ogni funzione *create* ha un singolo argomento **parent** che definisce a quale widget verrà aggiunto quello nuovo.

Ad esempio:

```
lv_obj_t * my_button1 = lv_button_create(lv_screen_active());
lv_obj_t * my_label1 = lv_label_create(my_button1);
```

Se un widget o una schermata non sono più necessari, possono essere rimossi chiamando `lv_obj_delete(my_button1)`

Per modificare le proprietà dei widget, è possibile utilizzare due set di funzioni:

- Funzioni `lv_obj_...()` per proprietà comuni, ad esempio `lv_obj_set_width()`, `lv_obj_add_style()`, ecc. Queste sono trattate in [Funzionalità Comuni dei Widget](#).
- Funzioni `lv_<widget_type>_...()` per proprietà specifiche del tipo, ad esempio `lv_label_set_text()`, `lv_slider_set_value()`, ecc.

Ecco un esempio che mostra anche alcune unità di misura *non-pixel* per le dimensioni:

```
lv_obj_t * my_button1 = lv_button_create(lv_screen_active());
/*Set parent-sized width, and content-sized height*/
lv_obj_set_size(my_button1, lv_pct(100), LV_SIZE_CONTENT);
/*Align to the right center with 20px offset horizontally*/
lv_obj_align(my_button1, LV_ALIGN_RIGHT_MID, -20, 0);

lv_obj_t * my_label1 = lv_label_create(my_button1);
lv_label_set_text_fmt(my_label1, "Click me!");
lv_obj_set_style_text_color(my_label1, lv_color_hex(0xff0000), 0);
/*Make the text red*/
```

Per visualizzare l'API completa di un widget, consultare la relativa documentazione in [Tutti i Widget](#) o controllare il relativo file header nel codice sorgente.

## 2.1.5 Eventi

Gli eventi vengono utilizzati per informare l'applicazione che si è verificato un problema con un widget. È possibile assegnare una o più callback a un Widget, che verranno richiamate quando il Widget verrà cliccato, rilasciato, trascinato, eliminato, ecc.

Una callback viene assegnata in questo modo:

```
lv_obj_add_event_cb(btn, my_btn_event_cb, LV_EVENT_CLICKED, NULL);
...
void my_btn_event_cb(lv_event_t * e)
{
    printf("Clicked\n");
}
```

È possibile utilizzare `LV_EVENT_ALL` al posto di `LV_EVENT_CLICKED` per richiamare la callback per tutti gli eventi.

Le callback degli eventi ricevono l'argomento `lv_event_t *` e contenente il codice dell'evento corrente e altre informazioni relative all'evento. Il codice dell'evento corrente può essere recuperato con:

```
lv_event_code_t code = lv_event_get_code(e);
```

Il Widget che ha attivato l'evento può essere recuperato con:

```
lv_obj_t * widget = lv_event_get_target_obj(e);
```

Si può scoprire tutto sugli eventi nella sezione [Eventi](#).

## 2.1.6 Parti e Stati

### Parti

I Widget sono costituiti da una o più *parti*. Ad esempio, un pulsante ha una sola *parte* chiamata `LV_PART_MAIN`. Tuttavia, uno *Slider* (`lv_slider`) ha `LV_PART_MAIN`, `LV_PART_INDICATOR` e `LV_PART_KNOB`.

Utilizzando le *parti* è possibile applicare diversi *stili* alle parti di un widget.

Consultare la documentazione del Widget per scoprire quali parti utilizza.

## Stati

I Widget possono trovarsi in una combinazione dei seguenti stati:

- *LV\_STATE\_DEFAULT*: Normale, stato rilasciato
- *LV\_STATE\_CHECKED*: Stato attivato o selezionato
- *LV\_STATE\_FOCUSED*: Focus tramite tastiera o encoder o clic tramite touchpad/mouse
- *LV\_STATE\_FOCUS\_KEY*: Focus tramite tastiera o encoder ma non tramite touchpad/mouse
- *LV\_STATE\_EDITED*: Modificato tramite encoder
- *LV\_STATE\_HOVERED*: Passaggio del mouse
- *LV\_STATE\_PRESSED*: Premuto
- *LV\_STATE\_SCROLLED*: In corso di scrolling
- *LV\_STATE\_DISABLED*: Disabilitato

Ad esempio, se si preme un Widget, questo passerà automaticamente agli stati *LV\_STATE\_FOCUSED* e *LV\_STATE\_PRESSED*. Quando lo si rilascia, lo stato *LV\_STATE\_PRESSED* verrà rimosso, mentre lo stato *LV\_STATE\_FOCUSED* rimarrà attivo.

Per verificare se un Widget si trova in un determinato stato, utilizzare *lv\_obj\_has\_state(widget, LV\_STATE\_...)*. Restituirà `true` se il widget si trova attualmente in quello stato.

Per aggiungere o rimuovere stati a livello di codice, utilizzare:

```
lv_obj_add_state(widget, LV_STATE_...);
lv_obj_remove_state(widget, LV_STATE_...);
```

## 2.1.7 Stili

### Inizializzazione degli stili

Gli stili sono contenuti in oggetti *lv\_style\_t*. Contengono proprietà come colore di sfondo, larghezza del bordo, font, ecc.

Gli stili possono essere aggiunti alla *Parte* e allo *Stato*. Solo il loro puntatore viene salvato nei Widget, quindi devono essere definiti come variabili statiche o globali.

Prima di utilizzare uno stile è necessario inizializzarlo con *lv\_style\_init(&style1)*. In seguito, è possibile aggiungere proprietà per configurare lo stile. Ad esempio:

```
static lv_style_t style1;
lv_style_init(&style1);
lv_style_set_bg_color(&style1, lv_color_hex(0xa03080));
lv_style_set_border_width(&style1, 2);
```

*Panoramica delle Proprietà di Stile* per ulteriori dettagli.

*Proprietà di Stile* per vedere l'elenco completo.

### Aggiungere stili ai widget

Successivamente è possibile aggiungere ai widget:

```
lv_obj_add_style(my_button1, &style1, 0); /*0 means add to the main part and default state*/
lv_obj_add_style(my_checkbox1, &style1, LV_STATE_DISABLED); /*Add to checkbox's disabled state*/
lv_obj_add_style(my_slider1, &style1, LV_PART_KNOB | LV_STATE_PRESSED); /*Add to the slider's knob pressed state*/
```

### Ereditarietà

Alcune proprietà (in particolare quelle relative al testo) possono essere ereditate. Ciò significa che se una proprietà non è impostata in un Widget, verrà cercata nei Widget genitore. Ad esempio, è possibile impostare il font una volta nello stile della schermata e tutto il testo su quella schermata lo erediterà per default, a meno che il font non sia specificato nel widget o in uno dei suoi genitori.

## Stili Locali

È anche possibile aggiungere proprietà di stile locali ai Widget. Questo crea uno stile che risiede all'interno del Widget e viene utilizzato solo da quel Widget:

```
lv_obj_set_style_bg_color(slider1, lv_color_hex(0x2080bb), LV_PART_INDICATOR | LV_STATE_PRESSED);
```

*Stili* per tutti i dettagli.

### 2.1.8 Soggetti e Osservatori

Soggetti e Osservatori sono strumenti potenti per creare facilmente data binding.

I soggetti sono variabili globali `lv_subject_t` che memorizzano valori interi, colori, stringhe, ecc.

L'interfaccia utente o l'applicazione possono sottoscrivere questi soggetti creando *callback dell'osservatore* che vengono notificati quando il soggetto cambia.

Anche un widget può sottoscrivere un soggetto. In questo modo, quando il widget viene eliminato, la sua sottoscrizione verrà automaticamente annullata.

Per alcuni widget, le funzioni di supporto semplificano la connessione ai soggetti. Ad esempio: `lv_slider_bind_value()`, `lv_label_bind_text()`.

In generale, l'utilizzo di *soggetti* e *osservatori* è un modo per collegare varie parti dell'interfaccia utente e farle reagire dinamicamente alle modifiche dei dati dell'applicazione, oppure consentire all'applicazione di reagire alle modifiche dell'interfaccia utente.

```
static void label_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_obj_t * label = lv_observer_get_target_obj(observer);
    lv_label_set_text_fmt(label, "Progress: %d", lv_subject_get_int(subject));
}

...
static lv_subject_t subject1;
lv_subject_init_int(&subject1, 10);

lv_obj_t * labell = lv_label_create(lv_screen_active());
/*lv_label_bind_text could have been used too*/
lv_subject_add_observer_obj(&subject1, label_observer_cb, labell, NULL);

lv_obj_t * slider1 = lv_slider_create(lv_screen_active());
lv_slider_bind_value(slider1, &subject1);
lv_obj_set_y(slider1, 30);

lv_subject_set_int(&subject1, 30);
```

Ulteriori approfondimenti nella pagina di documentazione degli *Osservatori*.

## 2.2 Esempi di Base

Di seguito sono riportati diversi esempi di base. Includono il codice applicativo che genera il Widget Tree necessario per far sì che LVGL esegua il rendering degli esempi mostrati. Ogni esempio presuppone che LVGL abbia subito una normale inizializzazione, ovvero che sia stato creato un oggetto `lv_display_t` e che quindi abbia un *Lo Screen Attivo*.

### 2.2.1 Un'etichetta *hello world* molto semplice

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

/**
 * Basic example to create a "Hello world" label
 */
void lv_example_get_started_1(void)
{
    /*Change the active screen's background color/
    lv_obj_set_style_bg_color(lv_screen_active(), lv_color_hex(0x003a57), LV_PART_MAIN);

    /*Create a white label, set its text and align it to the center*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello world");
    lv_obj_set_style_text_color(lv_screen_active(), lv_color_hex(0xffffffff), LV_PART_MAIN);
    lv_obj_align(label, LV_ALIGN_CENTER, 0, 0);
}
```

## 2.2.2 Un pulsante con un'etichetta e un evento di reazione al clic

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON

static void btn_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * btn = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        static uint8_t cnt = 0;
        cnt++;

        /*Get the first child of the button which is the label and change its text*/
        lv_obj_t * label = lv_obj_get_child(btn, 0);
        lv_label_set_text_fmt(label, "Button: %d", cnt);
    }
}

/**
 * Create a button with a label and react on click event.
 */
void lv_example_get_started_2(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active());           /*Add a button the current screen*/
    lv_obj_set_pos(btn, 10, 10);                                     /*Set its position*/
    lv_obj_set_size(btn, 120, 50);                                    /*Set its size*/
    lv_obj_add_event_cb(btn, btn_event_cb, LV_EVENT_ALL, NULL);     /*Assign a callback to the button*/

    lv_obj_t * label = lv_label_create(btn);                          /*Add a label to the button*/
    lv_label_set_text(label, "Button");                                /*Set the labels text*/
    lv_obj_center(label);
}

#endif
```

## 2.2.3 Creare stili da zero per i pulsanti

```
#include "../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

static lv_style_t style_btn;
static lv_style_t style_button_pressed;
static lv_style_t style_button_red;

static lv_color_t darken(const lv_color_filter_dsc_t * dsc, lv_color_t color, lv_opa_t opa)
{
    LV_UNUSED(dsc);
    return lv_color_darken(color, opa);
}

static void style_init(void)
{
    /*Create a simple button style*/
    lv_style_init(&style_btn);
    lv_style_set_radius(&style_btn, 10);
    lv_style_set_bg_opa(&style_btn, LV_OPA_COVER);
    lv_style_set_bg_color(&style_btn, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_bg_grad_color(&style_btn, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_bg_grad_dir(&style_btn, LV_GRAD_DIR_VERT);

    lv_style_set_border_color(&style_btn, lv_color_black());
    lv_style_set_border_opa(&style_btn, LV_OPA_20);
    lv_style_set_border_width(&style_btn, 2);

    lv_style_set_text_color(&style_btn, lv_color_black());

    /*Create a style for the pressed state.
     *Use a color filter to simply modify all colors in this state*/
    static lv_color_filter_dsc_t color_filter;
    lv_color_filter_dsc_init(&color_filter, darken);
    lv_style_init(&style_button_pressed);
    lv_style_set_color_filter_dsc(&style_button_pressed, &color_filter);
    lv_style_set_color_opa(&style_button_pressed, LV_OPA_20);

    /*Create a red style. Change only some colors.*/
    lv_style_init(&style_button_red);
    lv_style_set_bg_color(&style_button_red, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_color(&style_button_red, lv_palette_lighten(LV_PALETTE_RED, 3));
}

/**
 * Create styles from scratch for buttons.
 */
void lv_example_get_started_3(void)
{
    /*Initialize the style*/
    style_init();

    /*Create a button and use the new styles*/
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    /* Remove the styles coming from the theme
     * Note that size and position are also stored as style properties
     * so lv_obj_remove_style_all will remove the set size and position too */
    lv_obj_remove_style_all(btn);
    lv_obj_set_pos(btn, 10, 10);
    lv_obj_set_size(btn, 120, 50);
    lv_obj_add_style(btn, &style_btn, 0);
    lv_obj_add_style(btn, &style_button_pressed, LV_STATE_PRESSED);

    /*Add a label to the button*/
    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);

    /*Create another button and use the red style too*/
    lv_obj_t * btn2 = lv_button_create(lv_screen_active());
    lv_obj_remove_style_all(btn2);                                /*Remove the styles coming from the theme*/
    lv_obj_set_pos(btn2, 10, 80);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_size(btn2, 120, 50);
lv_obj_add_style(btn2, &style_btn, 0);
lv_obj_add_style(btn2, &style_button_red, 0);
lv_obj_add_style(btn2, &style_button_pressed, LV_STATE_PRESSED);
lv_obj_set_style_radius(btn2, LV_RADIUS_CIRCLE, 0); /*Add a local style too*/

label = lv_label_create(btn2);
lv_label_set_text(label, "Button 2");
lv_obj_center(label);
}

#endif

```

## 2.2.4 Creare uno slider e scriverne il valore su un'etichetta

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SLIDER

static lv_obj_t * label;

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target_obj(e);

    /*Refresh the text*/
    lv_label_set_text_fmt(label, "%" LV_PRIId32, lv_slider_get_value(slider));
    lv_obj_align_to(label, slider, LV_ALIGN_OUT_TOP_MID, 0, -15); /*Align top of the slider*/
}

/***
 * Create a slider and write its value on a label.
 */
void lv_example_get_started_4(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_set_width(slider, 200); /*Set the width*/
    lv_obj_center(slider); /*Align to the center of the parent (screen)*/
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL); /*Assign an event function*/

    /*Create a label above the slider*/
    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "0");
    lv_obj_align_to(label, slider, LV_ALIGN_OUT_TOP_MID, 0, -15); /*Align top of the slider*/
}
#endif

```

## 2.3 Prossimi Passi

Esistono diversi ottimi modi per approfondire la conoscenza di LVGL. Ecco un ordine consigliato di documenti da leggere e di spunti su cui esercitarsi nell'approfondire le proprie conoscenze:

1. Se non è stato già fatto, iniziare con la pagina [Introduzione](#) della documentazione. (5 minuti)
2. Consultate le [Online Demos](#) per vedere LVGL in azione. (3 minuti)
3. Se non è stato già fatto, leggere [Imparare le Basi](#) (sopra). (15 minuti)
4. Configura un [Simulatore su PC](#) LVGL. (10 minuti)
5. Consultare alcuni [Esempi](#) e il relativo codice.
6. [Aggiungere LVGL al progetto](#) o consultare i ready-to-use Projects.
7. Leggere le pagine di [Main Modules](#) per comprendere meglio la libreria. (2-3 ore)
8. Consultare la documentazione dei [Tutti i Widget](#) per vedere cosa è disponibile.
9. Se ci sono domande, visitare il [Forum](#).
10. Leggere la guida [Contribuire](#) per scoprire come contribuire a migliorare LVGL. (15 minuti)

# CAPITOLO 3

## Esempi

### 3.1 Per Iniziare

#### 3.1.1 Un'etichetta *hello world* molto semplice

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

/***
 * Basic example to create a "Hello world" label
 */
void lv_example_get_started_1(void)
{
    /*Change the active screen's background color*/
    lv_obj_set_style_bg_color(lv_screen_active(), lv_color_hex(0x003a57), LV_PART_MAIN);

    /*Create a white label, set its text and align it to the center*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello world");
    lv_obj_set_style_text_color(lv_screen_active(), lv_color_hex(0xffffffff), LV_PART_MAIN);
    lv_obj_align(label, LV_ALIGN_CENTER, 0, 0);
}

#endif
```

#### 3.1.2 Un pulsante con un'etichetta e un evento di reazione al clic

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON

static void btn_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * btn = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        static uint8_t cnt = 0;
        cnt++;

        /*Get the first child of the button which is the label and change its text*/
        lv_obj_t * label = lv_obj_get_child(btn, 0);
        lv_label_set_text_fmt(label, "Button: %d", cnt);
    }
}

/***
 * Create a button with a label and react on click event.
 */
void lv_example_get_started_2(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active());           /*Add a button the current screen*/
    lv_obj_set_pos(btn, 10, 10);                                     /*Set its position*/
    lv_obj_set_size(btn, 120, 50);                                    /*Set its size*/
    lv_obj_add_event_cb(btn, btn_event_cb, LV_EVENT_ALL, NULL);     /*Assign a callback to the button*/

    lv_obj_t * label = lv_label_create(btn);                          /*Add a label to the button*/
    lv_label_set_text(label, "Button");                               /*Set the labels text*/
    lv_obj_center(label);
}

#endif
```

### 3.1.3 Creare stili da zero per i pulsanti

```
#include "../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

static lv_style_t style_btn;
static lv_style_t style_button_pressed;
static lv_style_t style_button_red;

static lv_color_t darken(const lv_color_filter_dsc_t * dsc, lv_color_t color, lv_opa_t opa)
{
    LV_UNUSED(dsc);
    return lv_color_darken(color, opa);
}

static void style_init(void)
{
    /*Create a simple button style*/
    lv_style_init(&style_btn);
    lv_style_set_radius(&style_btn, 10);
    lv_style_set_bg_opa(&style_btn, LV_OPA_COVER);
    lv_style_set_bg_color(&style_btn, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_bg_grad_color(&style_btn, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_bg_grad_dir(&style_btn, LV_GRAD_DIR_VER);

    lv_style_set_border_color(&style_btn, lv_color_black());
    lv_style_set_border_opa(&style_btn, LV_OPA_20);
    lv_style_set_border_width(&style_btn, 2);

    lv_style_set_text_color(&style_btn, lv_color_black());

    /*Create a style for the pressed state.
     *Use a color filter to simply modify all colors in this state*/
    static lv_color_filter_dsc_t color_filter;
    lv_color_filter_dsc_init(&color_filter, darken);
    lv_style_init(&style_button_pressed);
    lv_style_set_color_filter_dsc(&style_button_pressed, &color_filter);
    lv_style_set_color_filter_opa(&style_button_pressed, LV_OPA_20);

    /*Create a red style. Change only some colors.*/
    lv_style_init(&style_button_red);
    lv_style_set_bg_color(&style_button_red, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_color(&style_button_red, lv_palette_lighten(LV_PALETTE_RED, 3));
}

/**
 * Create styles from scratch for buttons.
 */
void lv_example_get_started_3(void)
{
    /*Initialize the style*/
    style_init();

    /*Create a button and use the new styles*/
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    /* Remove the styles coming from the theme
     * Note that size and position are also stored as style properties
     * so lv_obj_remove_style_all will remove the set size and position too */
    lv_obj_remove_style_all(btn);
    lv_obj_set_pos(btn, 10, 10);
    lv_obj_set_size(btn, 120, 50);
    lv_obj_add_style(btn, &style_btn, 0);
    lv_obj_add_style(btn, &style_button_pressed, LV_STATE_PRESSED);

    /*Add a label to the button*/
    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);

    /*Create another button and use the red style too*/
    lv_obj_t * btn2 = lv_button_create(lv_screen_active());
    lv_obj_remove_style_all(btn2); /*Remove the styles coming from the theme*/
    lv_obj_set_pos(btn2, 10, 80);
    lv_obj_set_size(btn2, 120, 50);
    lv_obj_add_style(btn2, &style_btn, 0);
    lv_obj_add_style(btn2, &style_button_red, 0);
    lv_obj_add_style(btn2, &style_button_pressed, LV_STATE_PRESSED);
    lv_obj_set_style_radius(btn2, LV_RADIUS_CIRCLE, 0); /*Add a local style too*/

    label = lv_label_create(btn2);
    lv_label_set_text(label, "Button 2");
    lv_obj_center(label);
}

#endif
```

### 3.1.4 Creare uno slider e scriverne il valore su un'etichetta

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SLIDER

static lv_obj_t * label;

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target_obj(e);

    /*Refresh the text*/
    lv_label_set_text_fmt(label, "%" LV_PRId32, lv_slider_get_value(slider));
    lv_obj_align_to(label, slider, LV_ALIGN_OUT_TOP_MID, 0, -15); /*Align top of the slider*/
}

/**
 * Create a slider and write its value on a label.
 */
void lv_example_get_started_4(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_width(slider, 200);           /*Set the width*/
lv_obj_center(slider);                 /*Align to the center of the parent (screen)*/
lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL); /*Assign an event function*/

/*Create a label above the slider*/
label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "8");
lv_obj_align_to(label, slider, LV_ALIGN_OUT_TOP_MID, 0, -15); /*Align top of the slider*/
}

#endif

```

### 3.1.5 Semplice Gradiente Orizzontale

### 3.1.6 Gradiente (Obliquo) Lineare

### 3.1.7 Gradiente Radiale

### 3.1.8 Gradiente Conico

## 3.2 Stili

### 3.2.1 Stili Size

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Using the Size, Position and Padding style properties
 */
void lv_example_style_1(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_width(&style, 150);
    lv_style_set_height(&style, LV_SIZE_CONTENT);

    lv_style_set_pad_ver(&style, 20);
    lv_style_set_pad_left(&style, 5);

    lv_style_set_x(&style, lv_pct(50));
    lv_style_set_y(&style, 80);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label, "Hello");
}
#endif

```

### 3.2.2 Stili Background

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the background style properties
 */
void lv_example_style_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    static lv_grad_dsc_t grad;
    grad.dir = LV_GRAD_DIR_VER;
    grad.stops_count = 2;
    grad.stops[0].color = lv_palette_lighten(LV_PALETTE_GREY, 1);
    grad.stops[0].opa = LV_OPA_COVER;
    grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
    grad.stops[1].opa = LV_OPA_COVER;

    /*Shift the gradient to the bottom*/
    grad.stops[0].frac = 128;
    grad.stops[1].frac = 192;

    lv_style_set_bg_grad(&style, &grad);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif

```

### 3.2.3 Stili Border

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the border style properties
 */
void lv_example_style_3(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 10);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add border to the bottom+right*/
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style, 5);
    lv_style_set_border_opa(&style, LV_OPA_50);
    lv_style_set_border_side(&style, (lv_border_side_t)(LV_BORDER_SIDE_BOTTOM | LV_BORDER_SIDE_RIGHT));

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

### 3.2.4 Stili Outline

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the outline style properties
 */
void lv_example_style_4(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add outline*/
    lv_style_set_outline_width(&style, 2);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_outline_pad(&style, 8);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

### 3.2.5 Stili Shadow

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the Shadow style properties
 */
void lv_example_style_5(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add a shadow*/
    lv_style_set_shadow_width(&style, 55);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

### 3.2.6 Stili Image

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Using the Image style properties
 */
```

(continues on next page)

(continua dalla pagina precedente)

```
/*
void lv_example_style_6(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    lv_style_set_image_recolor(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_image_recolor_opa(&style, LV_OPA_50);
    lv_style_set_transform_rotation(&style, 300);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_image_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    LV_IMAGE_DECLARE(img_cogwheel_argb);
    lv_image_set_src(obj, &img_cogwheel_argb);

    lv_obj_center(obj);
}

#endif
```

### 3.2.7 Stili Arc

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_ARC

/**
 * Using the Arc style properties
 */
void lv_example_style_7(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_arc_color(&style, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_arc_width(&style, 4);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_arc_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

### 3.2.8 Stili Text

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

/**
 * Using the text style properties
 */
void lv_example_style_8(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 2));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_pad_all(&style, 10);

    lv_style_set_text_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_text_letter_space(&style, 5);
    lv_style_set_text_line_space(&style, 20);
    lv_style_set_text_decor(&style, LV_TEXT_DECOR_UNDERLINE);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_label_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_label_set_text(obj, "Text of\n"
                        "a label");

    lv_obj_center(obj);
}

#endif
```

### 3.2.9 Stili Line

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LINE

/**
 * Using the line style properties
 */
void lv_example_style_9(void)
{
    static lv_style_t style;
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_init(&style);

lv_style_set_line_color(&style, lv_palette_main(LV_PALETTE_GREY));
lv_style_set_line_width(&style, 6);
lv_style_set_line_rounded(&style, true);

/*Create an object with the new style*/
lv_obj_t * obj = lv_line_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);

static lv_point_precise_t p[] = {{10, 30}, {30, 50}, {100, 0}};
lv_line_set_points(obj, p, 3);

lv_obj_center(obj);

}

#endif

```

### 3.2.10 Transizione

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Creating a transition
 */
void lv_example_style_10(void)
{
    static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, LV_STYLE_BORDER_COLOR, LV_STYLE_BORDER_WIDTH, 0};

    /* A default transition
     * Make it fast (100ms) and start with some delay (200 ms)*/
    static lv_style_transition_dsc_t trans_def;
    lv_style_transition_dsc_init(&trans_def, props, lv_anim_path_linear, 100, 200, NULL);

    /* A special transition when going to pressed state
     * Make it slow (500 ms) but start without delay*/
    static lv_style_transition_dsc_t trans_pr;
    lv_style_transition_dsc_init(&trans_pr, props, lv_anim_path_linear, 500, 0, NULL);

    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_transition(&style_def, &trans_def);

    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_bg_color(&style_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_border_width(&style_pr, 6);
    lv_style_set_border_color(&style_pr, lv_palette_darken(LV_PALETTE_RED, 3));
    lv_style_set_transition(&style_pr, &trans_pr);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style_def, 0);
    lv_obj_add_style(obj, &style_pr, LV_STATE_PRESSED);

    lv_obj_center(obj);
}
#endif

```

### 3.2.11 Utilizzo di stili multipli

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Using multiple styles
 */
void lv_example_style_11(void)
{
    /*A base style*/
    static lv_style_t style_base;
    lv_style_init(&style_base);
    lv_style_set_bg_color(&style_base, lv_palette_main(LV_PALETTE_LIGHT_BLUE));
    lv_style_set_border_color(&style_base, lv_palette_darken(LV_PALETTE_LIGHT_BLUE, 3));
    lv_style_set_border_width(&style_base, 2);
    lv_style_set_radius(&style_base, 10);
    lv_style_set_shadow_width(&style_base, 10);
    lv_style_set_shadow_offset_y(&style_base, 5);
    lv_style_set_shadow_opa(&style_base, LV_OPA_50);
    lv_style_set_text_color(&style_base, lv_color_white());
    lv_style_set_width(&style_base, 100);
    lv_style_set_height(&style_base, LV_SIZE_CONTENT);

    /*Set only the properties that should be different*/
    static lv_style_t style_warning;
    lv_style_init(&style_warning);
    lv_style_set_bg_color(&style_warning, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_border_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW, 3));
    lv_style_set_text_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW, 4));

    /*Create an object with the base style only*/
    lv_obj_t * obj_base = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj_base, &style_base, 0);
    lv_obj_align(obj_base, LV_ALIGN_LEFT_MID, 20, 0);

    lv_obj_t * label = lv_label_create(obj_base);
    lv_label_set_text(label, "Base");
    lv_obj_center(label);

    /*Create another object with the base style and earnings style too*/
    lv_obj_t * obj_warning = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj_warning, &style_base, 0);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_add_style(obj_warning, &style_warning, 0);
lv_obj_align(obj_warning, LV_ALIGN_RIGHT_MID, -20, 0);

label = lv_label_create(obj_warning);
lv_label_set_text(label, "Warning");
lv_obj_center(label);
}

#endif

```

### 3.2.12 Stili Locali

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Local styles
 */
void lv_example_style_12(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style, lv_palette_lighten(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style, 3);

    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    /*Overwrite the background color locally*/
    lv_obj_set_style_bg_color(obj, lv_palette_main(LV_PALETTE_ORANGE), LV_PART_MAIN);

    lv_obj_center(obj);
}
#endif

```

### 3.2.13 Aggiunta di stili a parti e a stati

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Add styles to parts and states
 */
void lv_example_style_13(void)
{
    static lv_style_t style_indic;
    lv_style_init(&style_indic);
    lv_style_set_bg_color(&style_indic, lv_palette_lighten(LV_PALETTE_RED, 3));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_HOR);

    static lv_style_t style_indic_pr;
    lv_style_init(&style_indic_pr);
    lv_style_set_shadow_color(&style_indic_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_shadow_width(&style_indic_pr, 10);
    lv_style_set_shadow_spread(&style_indic_pr, 3);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_slider_create(lv_screen_active());
    lv_obj_add_style(obj, &style_indic, LV_PART_INDICATOR);
    lv_obj_add_style(obj, &style_indic_pr, LV_PART_INDICATOR | LV_STATE_PRESSED);
    lv_slider_set_value(obj, 70, LV_ANIM_OFF);
    lv_obj_center(obj);
}
#endif

```

### 3.2.14 Estensione del tema corrente

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE
#include "../lvgl_private.h"

static lv_style_t style_btn;

/*Will be called when the styles of the base theme are already added
to add new styles*/
static void new_theme_apply_cb(lv_theme_t * th, lv_obj_t * obj)
{
    LV_UNUSED(th);

    if(lv_obj_check_type(obj, &lv_button_class)) {
        lv_obj_add_style(obj, &style_btn, 0);
    }
}

static void new_theme_init_and_set(void)
{
    /*Initialize the styles*/
    lv_style_init(&style_btn);
    lv_style_set_bg_color(&style_btn, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style_btn, lv_palette_darken(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style_btn, 3);

    /*Initialize the new theme from the current theme*/
    lv_theme_t * th_act = lv_display_get_theme(NULL);
    static lv_theme_t th_new;

```

(continues on next page)

(continua dalla pagina precedente)

```

th_new = *th_act;

/*Set the parent theme and the style apply callback for the new theme*/
lv_theme_set_parent(&th_new, th_act);
lv_theme_set_apply_cb(&th_new, new_theme_apply_cb);

/*Assign the new theme to the current display*/
lv_display_set_theme(NULL, &th_new);
}

/**
 * Extending the current theme
 */
void lv_example_style_14(void)
{
    lv_obj_t * btn;
    lv_obj_t * label;

    btn = lv_button_create(lv_screen_active());
    lv_obj_align(btn, LV_ALIGN_TOP_MID, 0, 20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Original theme");

    new_theme_init_and_set();

    btn = lv_button_create(lv_screen_active());
    lv_obj_align(btn, LV_ALIGN_BOTTOM_MID, 0, -20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "New theme");
}

#endif

```

### 3.2.15 Opacità e trasformazioni

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON && LV_USE_LABEL

/**
 * Opacity and Transformations
 */
void lv_example_style_15(void)
{
    lv_obj_t * btn;
    lv_obj_t * label;

    /*Normal button*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -70);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Normal");
    lv_obj_center(label);

    /*Set opacity
     *The button and the label is rendered to a layer first and that layer is blended*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_set_style_opa(btn, LV_OPA_50, 0);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 0);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Opa:50%");
    lv_obj_center(label);

    /*Set transformations
     *The button and the label is rendered to a layer first and that layer is transformed*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_set_style_transform_rotation(btn, 150, 0);      /*15 deg*/
    lv_obj_set_style_transform_scale(btn, 256 + 64, 0);   /*1.25x*/
    lv_obj_set_style_transform_pivot_x(btn, 50, 0);
    lv_obj_set_style_transform_pivot_y(btn, 20, 0);
    lv_obj_set_style_opa(btn, LV_OPA_50, 0);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 70);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Transf.");
    lv_obj_center(label);
}

#endif

```

### 3.2.16 Manopola metallica con gradiente conico

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/**
 * Simulate metallic knob using conical gradient
 * For best effect set LV_GRADIENT_MAX_STOPS to 8 or at least 3
 */
void lv_example_style_16(void)
{
#if LV_GRADIENT_MAX_STOPS >= 8
    static const lv_color_t grad_colors[8] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0xff, 0xff, 0xff),
        LV_COLOR_MAKE(0xfa, 0xfa, 0xfa),

```

(continues on next page)

(continua dalla pagina precedente)

```

LV_COLOR_MAKE(0x79, 0x79, 0x79),
LV_COLOR_MAKE(0x48, 0x48, 0x48),
LV_COLOR_MAKE(0x4b, 0x4b, 0x4b),
LV_COLOR_MAKE(0x70, 0x70, 0x70),
LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
};

#elif LV_GRADIENT_MAX_STOPS >= 3
static const lv_color_t grad_colors[3] = {
    LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
    LV_COLOR_MAKE(0xff, 0xff, 0xff),
    LV_COLOR_MAKE(0x79, 0x79, 0x79),
};
#else
static const lv_color_t grad_colors[2] = {
    LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
    LV_COLOR_MAKE(0x79, 0x79, 0x79),
};
#endif

/*Create a style with gradient background and shadow*/
static lv_style_t style;
lv_style_init(&style);
lv_style_set_radius(&style, 500);
lv_style_set_bg_opa(&style, LV_OPA_COVER);
lv_style_set_shadow_color(&style, lv_color_black());
lv_style_set_shadow_width(&style, 50);
lv_style_set_shadow_offset_x(&style, 20);
lv_style_set_shadow_offset_y(&style, 20);
lv_style_set_shadow_opa(&style, LV_OPA_50);

/*First define a color gradient. In this example we use a gray color map with random values.*/
static lv_grad_desc_t grad;

lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

/*Make a conical gradient with the center in the middle of the object*/
#if LV_GRADIENT_MAX_STOPS >= 8
lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 0, 120, LV_GRAD_EXTEND_REFLECT);
#elif LV_GRADIENT_MAX_STOPS >= 3
lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 125, LV_GRAD_EXTEND_REFLECT);
#else
lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 110, LV_GRAD_EXTEND_REFLECT);
#endif

/*Set gradient as background*/
lv_style_set_bg_grad(&style, &grad);

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_set_size(obj, 200, 200);
lv_obj_center(obj);
}

#else

void lv_example_style_16(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

### 3.2.17 Gradiente radiale come sfondo

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Using radial gradient as background
 */
void lv_example_style_17(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x9b, 0x18, 0x42),
        LV_COLOR_MAKE(0x00, 0x00, 0x00),
    };

    int32_t width = lv_display_get_horizontal_resolution(NULL);
    int32_t height = lv_display_get_vertical_resolution(NULL);

    static lv_style_t style;
    lv_style_init(&style);

    /*First define a color gradient. In this example we use a purple to black color map.*/
    static lv_grad_desc_t grad;

    lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

    /*Make a radial gradient with the center in the middle of the object, extending to the farthest corner*/
    lv_grad_radial_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, LV_GRAD_RIGHT, LV_GRAD_BOTTOM, LV_GRAD_EXTEND_PAD);

    /*Set gradient as background*/
    lv_style_set_bg_grad(&style, &grad);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_set_size(obj, width, height);
    lv_obj_center(obj);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

}

#ifndef LV_USE_DRAW_SW_COMPLEX_GRADIENTS
void lv_example_style_17(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

### 3.2.18 Gradienti per lo sfondo del pulsante

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#ifndef LV_USE_DRAW_SW_COMPLEX_GRADIENTS
void lv_example_style_18(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x26, 0xa0, 0xda),
        LV_COLOR_MAKE(0x31, 0x47, 0x55),
    };

    /*Create a linear gradient going from the top left corner to the bottom at an angle, with reflected color map*/
    static lv_style_t style_with_linear_gradient_bg;
    static lv_grad_dsc_t linear_gradient_dsc;      /*NOTE: the gradient descriptor must be static or global variable!*/
    lv_style_init(&style_with_linear_gradient_bg);
    lv_grad_init_stops(&linear_gradient_dsc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_linear_init(&linear_gradient_dsc, lv_pct(0), lv_pct(0), lv_pct(20), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_linear_gradient_bg, &linear_gradient_dsc);
    lv_style_set_bg_opa(&style_with_linear_gradient_bg, LV_OPA_COVER);

    /*Create a radial gradient with the center in the top left 1/3rd of the object, extending to the bottom right corner, with reflected color map*/
    static lv_style_t style_with_radial_gradient_bg;
    static lv_grad_dsc_t radial_gradient_dsc;      /*NOTE: the gradient descriptor must be static or global variable!*/
    lv_style_init(&style_with_radial_gradient_bg);
    lv_grad_init_stops(&radial_gradient_dsc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_radial_init(&radial_gradient_dsc, lv_pct(30), lv_pct(30), lv_pct(100), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_radial_gradient_bg, &radial_gradient_dsc);
    lv_style_set_bg_opa(&style_with_radial_gradient_bg, LV_OPA_COVER);

    /*Create buttons with different gradient styles*/

    lv_obj_t * btn;
    lv_obj_t * label;

    /*Simple horizontal gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
    lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
    lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_HOR, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -100);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Horizontal");
    lv_obj_center(label);

    /*Simple vertical gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
    lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
    lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_VER, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -40);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Vertical");
    lv_obj_center(label);

    /*Complex linear gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_add_style(btn, &style_with_linear_gradient_bg, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Linear");
    lv_obj_center(label);

    /*Complex radial gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_add_style(btn, &style_with_radial_gradient_bg, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 80);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Radial");
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * label = lv_label_create(lv_screen_active());
lv_obj_set_width(label, LV_PCT(80));
lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

### 3.2.19 Test tra stile di ricolorazione o sfondo modale pieno

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SLIDER && LV_USE_LOG

/**
 * Test between a full background modal and a recolor modal
 */
void lv_example_style_19(void)
{
    /*Add lv_example_style_11 as background*/
    lv_example_style_11();

    /* Set to 1 to enable recolor overlay instead of solid background */
#if 0
    /* Apply a screen-wide tint using recolor (efficient overlay).
     * This modifies the visual appearance by blending a semi-transparent color
     * over existing content without creating additional objects.
     * It's lighter on performance compared to a full-size background object. */
    lv_obj_set_style_recolor(lv_screen_active(), lv_color_black(), 0);
    lv_obj_set_style_recolor_opa(lv_screen_active(), LV_OPA_50, 0);
#else
    /* Simulate a modal background by setting a semi-transparent black background
     * on lv_layer_top(), the highest built-in layer.
     * This method creates a new full-screen object and can consume more resources
     * compared to recolor, especially when using images or gradients. */
    lv_obj_set_style_bg_color(lv_layer_top(), lv_color_black(), 0);
    lv_obj_set_style_bg_opa(lv_layer_top(), LV_OPA_50, 0);
#endif

    lv_obj_t * obj = lv_slider_create(lv_layer_top());
    lv_obj_center(obj);

    lv_refr_now(NULL); /*Update layouts and render*/

    lv_obj_invalidate(lv_screen_active());

    uint32_t t = lv_tick_get();
    lv_refr_now(NULL); /*Render only*/
    LV_LOG_USER("%" LV_PRIu32 " ms\n", lv_tick_elaps(t));
}
#endif

```

## 3.3 Animazioni

### 3.3.1 Avviare animazione su un evento

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) var, v);
}

static void sw_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *) lv_event_get_user_data(e);

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), 100);
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_overshoot);
        lv_anim_start(&a);
    }
    else {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), -lv_obj_get_width(label));
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_ease_in);
        lv_anim_start(&a);
    }
}

/** Start animation on an event
 */
void lv_example_anim_1(void)
{

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "Hello animations!");
lv_obj_set_pos(label, 100, 10);

lv_obj_t * sw = lv_switch_create(lv_screen_active());
lv_obj_center(sw);
lv_obj_add_state(sw, LV_STATE_CHECKED);
lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_VALUE_CHANGED, label);
}

#endif

```

### 3.3.2 Riproduzione di animazione

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) var, v);
}

static void anim_size_cb(void * var, int32_t v)
{
    lv_obj_set_size((lv_obj_t *) var, v, v);
}

/**
 * Create a playback animation
 */
void lv_example_anim_2(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_remove_flag(obj, LV_FLAG_SCROLLABLE);
    lv_obj_set_style_bg_color(obj, lv_palette_main(LV_PALETTE_RED), 0);
    lv_obj_set_style_radius(obj, LV_RADIUS_CIRCLE, 0);

    lv_obj_align(obj, LV_ALIGN_LEFT_MID, 10, 0);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, obj);
    lv_anim_set_values(&a, 10, 50);
    lv_anim_set_duration(&a, 1000);
    lv_anim_set_reverse_delay(&a, 100);
    lv_anim_set_reverse_duration(&a, 300);
    lv_anim_set_repeat_delay(&a, 500);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_path_cb(&a, lv_anim_path_ease_in_out);

    lv_anim_set_exec_cb(&a, anim_size_cb);
    lv_anim_start(&a);
    lv_anim_set_exec_cb(&a, anim_x_cb);
    lv_anim_set_values(&a, 10, 240);
    lv_anim_start(&a);
}

#endif

```

### 3.3.3 Animazione di cubica di Bézier

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SLIDER && LV_USE_CHART && LV_USE_BUTTON && LV_USE_GRID

/**
 * the example show the use of cubic-bezier3 in animation.
 * the control point P1,P2 of cubic-bezier3 can be adjusted by slider.
 * and the chart shows the cubic-bezier3 in real time. you can click
 * run button see animation in current point of cubic-bezier3.
 */

#define CHART_POINTS_NUM 256

struct {
    lv_obj_t * anim_obj;
    lv_obj_t * chart;
    lv_chart_series_t * ser1;
    lv_obj_t * p1_slider;
    lv_obj_t * p1_label;
    lv_obj_t * p2_slider;
    lv_obj_t * p2_label;
    lv_obj_t * run_btn;
    int32_t p1;
    int32_t p2;
    lv_anim_t a;
} ginfo;

static int32_t anim_path_bezier3_cb(const lv_anim_t * a);
static void refer_chart_cubic_bezier(void);
static void run_button_event_handler(lv_event_t * e);
static void slider_event_cb(lv_event_t * e);
static void page_obj_init(lv_obj_t * par);
static void anim_x_cb(void * var, int32_t v);

/**
 * create an animation
 */
void lv_example_anim_3(void)
{
    static int32_t col_dsc[] = {LV_GRID_FR(1), 200, LV_GRID_FR(1), LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {30, 10, 10, LV_GRID_FR(1), LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * cont = lv_obj_create(lv_screen_active());
lv_obj_set_style_pad_all(cont, 2, LV_PART_MAIN);
lv_obj_set_style_pad_column(cont, 10, LV_PART_MAIN);
lv_obj_set_style_pad_row(cont, 10, LV_PART_MAIN);
lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
lv_obj_set_size(cont, 320, 240);
lv_obj_center(cont);
page_obj_init(cont);

lv_anim_init(&ginfo.a);
lv_anim_set_var(&ginfo.a, ginfo.anim_obj);
int32_t end = lv_obj_get_style_width(cont, LV_PART_MAIN) -
    lv_obj_get_style_width(ginfo.anim_obj, LV_PART_MAIN) - 10;
lv_anim_set_values(&ginfo.a, 5, end);
lv_anim_set_duration(&ginfo.a, 2000);
lv_anim_set_exec_cb(&ginfo.a, anim_x_cb);
lv_anim_set_path_cb(&ginfo.a, anim_path_bezier3_cb);
refer_chart_cubic_bezier();

static int32_t anim_path_bezier3_cb(const lv_anim_t * a)
{
    int32_t t = lv_map(a->act_time, 0, a->duration, 0, 1024);
    int32_t step = lv_bezier3(t, 0, ginfo.p1, ginfo.p2, 1024);
    int32_t new_value;
    new_value = step * (a->end_value - a->start_value);
    new_value = new_value >> 10;
    new_value += a->start_value;
    return new_value;
}

static void refer_chart_cubic_bezier(void)
{
    for(uint16_t i = 0; i <= CHART_POINTS_NUM; i++) {
        int32_t t = i * (1024 / CHART_POINTS_NUM);
        int32_t step = lv_bezier3(t, 0, ginfo.p1, ginfo.p2, 1024);
        lv_chart_set_series_value_by_id2(ginfo.chart, ginfo.ser1, i, t, step);
    }
    lv_chart_refresh(ginfo.chart);
}

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_style_translate_x((lv_obj_t *)var, v, LV_PART_MAIN);
}

static void run_button_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_CLICKED) {
        lv_anim_start(&ginfo.a);
    }
}

static void slider_event_cb(lv_event_t * e)
{
    char buf[16];
    lv_obj_t * label;
    lv_obj_t * slider = lv_event_get_target_obj(e);

    if(slider == ginfo.p1_slider) {
        label = ginfo.p1_label;
        ginfo.p1 = lv_slider_get_value(slider);
        lv_snprintf(buf, sizeof(buf), "p1:%d", ginfo.p1);
    } else {
        label = ginfo.p2_label;
        ginfo.p2 = lv_slider_get_value(slider);
        lv_snprintf(buf, sizeof(buf), "p2:%d", ginfo.p2);
    }

    lv_label_set_text(label, buf);
    refer_chart_cubic_bezier();
}

static void page_obj_init(lv_obj_t * par)
{
    ginfo.anim_obj = lv_obj_create(par);
    lv_obj_set_size(ginfo.anim_obj, 30, 30);
    lv_obj_set_align(ginfo.anim_obj, LV_ALIGN_TOP_LEFT);
    lv_obj_remove_flag(ginfo.anim_obj, LV_OBJ_FLAG_SCROLLABLE);
    lv_obj_set_style_bg_color(ginfo.anim_obj, lv_palette_main(LV_PALETTE_RED), LV_PART_MAIN);
    lv_obj_set_grid_cell(ginfo.anim_obj, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 0, 1);

    ginfo.p1_label = lv_label_create(par);
    ginfo.p2_label = lv_label_create(par);
    lv_label_set_text(ginfo.p1_label, "p1:0");
    lv_label_set_text(ginfo.p2_label, "p2:0");
    lv_obj_set_grid_cell(ginfo.p1_label, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 1, 1);
    lv_obj_set_grid_cell(ginfo.p2_label, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 2, 1);

    ginfo.p1_slider = lv_slider_create(par);
    ginfo.p2_slider = lv_slider_create(par);
    lv_slider_set_range(ginfo.p1_slider, 0, 1024);
    lv_slider_set_range(ginfo.p2_slider, 0, 1024);
    lv_obj_set_style_pad_all(ginfo.p1_slider, 2, LV_PART_KNOB);
    lv_obj_add_event_cb(ginfo.p1_slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_add_event_cb(ginfo.p2_slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_grid_cell(ginfo.p1_slider, LV_GRID_ALIGN_STRETCH, 1, 1, LV_GRID_ALIGN_START, 1, 1);
    lv_obj_set_grid_cell(ginfo.p2_slider, LV_GRID_ALIGN_STRETCH, 1, 1, LV_GRID_ALIGN_START, 2, 1);

    ginfo.run_btn = lv_button_create(par);
    lv_obj_add_event_cb(ginfo.run_btn, run_button_event_handler, LV_EVENT_CLICKED, NULL);
    lv_obj_t * btn_label = lv_label_create(ginfo.run_btn);
    lv_label_set_text(btn_label, LV_SYMBOL_PLAY);
    lv_obj_center(btn_label);
    lv_obj_set_grid_cell(ginfo.run_btn, LV_GRID_ALIGN_STRETCH, 2, 1, LV_GRID_ALIGN_STRETCH, 1, 2);

    ginfo.chart = lv_chart_create(par);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_style_pad_all(ginfo.chart, 0, LV_PART_MAIN);
lv_obj_set_style_size(ginfo.chart, 0, 0, LV_PART_INDICATOR);
lv_chart_set_type(ginfo.chart, LV_CHART_TYPE_SCATTER);
ginfo.ser1 = lv_chart_add_series(ginfo.chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
lv_chart_set_axis_range(ginfo.chart, LV_CHART_AXIS_PRIMARY_Y, 0, 1024);
lv_chart_set_axis_range(ginfo.chart, LV_CHART_AXIS_PRIMARY_X, 0, 1024);
lv_chart_set_point_count(ginfo.chart, CHART_POINTS_NUM);
lv_obj_set_grid_cell(ginfo.chart, LV_GRID_ALIGN_STRETCH, 0, 3, LV_GRID_ALIGN_STRETCH, 3, 1);
}

#endif

```

### 3.3.4 Pausa animazione

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) var, v);
}

static void timer_cb(lv_timer_t * timer)
{
    lv_anim_t * anim = (lv_anim_t *) lv_timer_get_user_data(timer);
    lv_anim_pause_for(anim, 1000);
    lv_timer_delete(timer);
}
static void sw_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *) lv_event_get_user_data(e);

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), 100);
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_overshoot);
        lv_timer_create(timer_cb, 200, lv_anim_start(&a));
    }
    else {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), -lv_obj_get_width(label));
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_ease_in);
        lv_timer_create(timer_cb, 200, lv_anim_start(&a));
    }
}

/***
 * Start animation on an event
 */
void lv_example_anim_4(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello animations!");
    lv_obj_set_pos(label, 100, 10);

    lv_obj_t * sw = lv_switch_create(lv_screen_active());
    lv_obj_center(sw);
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_VALUE_CHANGED, label);
}
#endif

```

### 3.3.5 Timeline animazione

```

#include "../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

static const int32_t obj_width = 90;
static const int32_t obj_height = 70;

static void set_width(lv_anim_t * var, int32_t v)
{
    lv_obj_set_width((lv_obj_t *) var->var, v);
}

static void set_height(lv_anim_t * var, int32_t v)
{
    lv_obj_set_height((lv_obj_t *) var->var, v);
}

static void set_slider_value(lv_anim_t * var, int32_t v)
{
    lv_slider_set_value((lv_obj_t *) var->var, v, LV_ANIM_OFF);
}

static void btn_start_event_handler(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_current_target_obj(e);
    lv_anim_timeline_t * anim_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);
    bool reverse = lv_obj_has_state(btn, LV_STATE_CHECKED);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_anim_timeline_set_reverse(animated_timeline, reverse);
    lv_anim_timeline_start(animated_timeline);
}

static void btn_pause_event_handler(lv_event_t * e)
{
    lv_anim_timeline_t * animated_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);
    lv_anim_timeline_pause(animated_timeline);
}

static void slider_prg_event_handler(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_current_target_obj(e);
    lv_anim_timeline_t * animated_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);
    int32_t progress = lv_slider_get_value(slider);
    lv_anim_timeline_set_progress(animated_timeline, (uint16_t) progress);
}

/***
 * Create an animation timeline
 */
void lv_example_anim_timeline_1(void)
{
    /* Create anim timeline */
    lv_anim_timeline_t * animated_timeline = lv_anim_timeline_create();

    lv_obj_t * par = lv_screen_active();
    lv_obj_set.flex_flow(par, LV_FLEX_FLOW_ROW);
    lv_obj_set.flex_align(par, LV_FLEX_ALIGN_SPACE_AROUND, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    /* create btn_start */
    lv_obj_t * btn_start = lv_button_create(par);
    lv_obj_add_event_cb(btn_start, btn_start_event_handler, LV_EVENT_VALUE_CHANGED, animated_timeline);
    lv_obj_add_flag(btn_start, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_add_flag(btn_start, LV_OBJ_FLAG_CHECKABLE);
    lv_obj_align(btn_start, LV_ALIGN_TOP_MID, -100, 20);

    lv_obj_t * label_start = lv_label_create(btn_start);
    lv_label_set_text(label_start, "Start");
    lv_obj_center(label_start);

    /* create btn_pause */
    lv_obj_t * btn_pause = lv_button_create(par);
    lv_obj_add_event_cb(btn_pause, btn_pause_event_handler, LV_EVENT_CLICKED, animated_timeline);
    lv_obj_add_flag(btn_pause, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(btn_pause, LV_ALIGN_TOP_MID, 100, 20);

    lv_obj_t * label_pause = lv_label_create(btn_pause);
    lv_label_set_text(label_pause, "Pause");
    lv_obj_center(label_pause);

    /* create slider_prg */
    lv_obj_t * slider_prg = lv_slider_create(par);
    lv_obj_add_event_cb(slider_prg, slider_prg_event_handler, LV_EVENT_VALUE_CHANGED, animated_timeline);
    lv_obj_add_flag(slider_prg, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(slider_prg, LV_ALIGN_BOTTOM_MID, 0, -20);
    lv_slider_set_range(slider_prg, 0, LV_ANIM_TIMELINE_PROGRESS_MAX);

    /* create 3 objects */
    lv_obj_t * obj1 = lv_obj_create(par);
    lv_obj_set_size(obj1, obj_width, obj_height);
    lv_obj_set_scrollbar_mode(obj1, LV_SCROLLBAR_MODE_OFF);

    lv_obj_t * obj2 = lv_obj_create(par);
    lv_obj_set_size(obj2, obj_width, obj_height);
    lv_obj_set_scrollbar_mode(obj2, LV_SCROLLBAR_MODE_OFF);

    lv_obj_t * obj3 = lv_obj_create(par);
    lv_obj_set_size(obj3, obj_width, obj_height);
    lv_obj_set_scrollbar_mode(obj3, LV_SCROLLBAR_MODE_OFF);

    /* anim-slider */
    lv_anim_t a_slider;
    lv_anim_init(&a_slider);
    lv_anim_set_var(&a_slider, slider_prg);
    lv_anim_set_values(&a_slider, 0, LV_ANIM_TIMELINE_PROGRESS_MAX);
    lv_anim_set_custom_exec_cb(&a_slider, set_slider_value);
    lv_anim_set_path_cb(&a_slider, lv_anim_path_linear);
    lv_anim_set_duration(&a_slider, 700);

    /* anim-obj1 */
    lv_anim_t a1;
    lv_anim_init(&a1);
    lv_anim_set_var(&a1, obj1);
    lv_anim_set_values(&a1, 0, obj_width);
    lv_anim_set_custom_exec_cb(&a1, set_width);
    lv_anim_set_path_cb(&a1, lv_anim_path_overshoot);
    lv_anim_set_duration(&a1, 300);

    lv_anim_t a2;
    lv_anim_init(&a2);
    lv_anim_set_var(&a2, obj1);
    lv_anim_set_values(&a2, 0, obj_height);
    lv_anim_set_custom_exec_cb(&a2, set_height);
    lv_anim_set_path_cb(&a2, lv_anim_path_ease_out);
    lv_anim_set_duration(&a2, 300);

    /* anim-obj2 */
    lv_anim_t a3;
    lv_anim_init(&a3);
    lv_anim_set_var(&a3, obj2);
    lv_anim_set_values(&a3, 0, obj_width);
    lv_anim_set_custom_exec_cb(&a3, set_width);
    lv_anim_set_path_cb(&a3, lv_anim_path_overshoot);
    lv_anim_set_duration(&a3, 300);

    lv_anim_t a4;
    lv_anim_init(&a4);
    lv_anim_set_var(&a4, obj2);
    lv_anim_set_values(&a4, 0, obj_height);
    lv_anim_set_custom_exec_cb(&a4, set_height);
    lv_anim_set_path_cb(&a4, lv_anim_path_ease_out);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_set_duration(&a4, 300);

/* anim-obj3 */
lv_anim_t a5;
lv_anim_init(&a5);
lv_anim_set_var(&a5, obj3);
lv_anim_set_values(&a5, 0, obj_width);
lv_anim_set_custom_exec_cb(&a5, set_width);
lv_anim_set_path_cb(&a5, lv_anim_path_overshoot);
lv_anim_set_duration(&a5, 300);

lv_anim_t a6;
lv_anim_init(&a6);
lv_anim_set_var(&a6, obj3);
lv_anim_set_values(&a6, 0, obj_height);
lv_anim_set_custom_exec_cb(&a6, set_height);
lv_anim_set_path_cb(&a6, lv_anim_path_ease_out);
lv_anim_set_duration(&a6, 300);

/* add animations to timeline */
lv_anim_timeline_add(anim_timeline, 0, &a_slider);
lv_anim_timeline_add(anim_timeline, 0, &a1);
lv_anim_timeline_add(anim_timeline, 0, &a2);
lv_anim_timeline_add(anim_timeline, 200, &a3);
lv_anim_timeline_add(anim_timeline, 200, &a4);
lv_anim_timeline_add(anim_timeline, 400, &a5);
lv_anim_timeline_add(anim_timeline, 400, &a6);

lv_anim_timeline_set_progress(anim_timeline, LV_ANIM_TIMELINE_PROGRESS_MAX);
}

#endif

```

## 3.4 Eventi

### 3.4.1 Evento clic pulsante

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    LV_LOG_USER("Clicked");

    static uint32_t cnt = 1;
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    lv_label_set_text_fmt(label, "%" LV_PRIu32, cnt);
    cnt++;
}

/**
 * Add click event to a button
 */
void lv_example_event_click(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);

    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Click me!");
    lv_obj_center(label);
}
#endif

```

### 3.4.2 Sequenze di clic

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

static void short_click_event_cb(lv_event_t * e)
{
    LV_LOG_USER("Short clicked");

    lv_obj_t * info_label = (lv_obj_t *) lv_event_get_user_data(e);
    lv_indev_t * index = (lv_indev_t *) lv_event_get_param(e);
    uint8_t cnt = lv_indev_get_short_click_streak(index);
    lv_label_set_text_fmt(info_label, "Short click streak: %u", cnt);
}

static void streak_event_cb(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    const char * text = (const char *) lv_event_get_user_data(e);
    lv_label_set_text(label, text);
}

void lv_example_event_streak(void)
{
    lv_obj_t * info_label = lv_label_create(lv_screen_active());
    lv_label_set_text(info_label, "No events yet");

    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);
    lv_obj_add_event_cb(btn, short_click_event_cb, LV_EVENT_SHORT_CLICKED, info_label);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_SINGLE_CLICKED, (void *) "Single clicked");
lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_DOUBLE_CLICKED, (void *) "Double clicked");
lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_TRIPLE_CLICKED, (void *) "Triple clicked");

lv_obj_t * label = lv_label_create(btn);
lv_label_set_text(label, "Click me!");
lv_obj_center(label);

}

#endif

```

### 3.4.3 Gestione di eventi multipli

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * label = (lv_obj_t *) lv_event_get_user_data(e);

    switch(code) {
        case LV_EVENT_PRESSED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_PRESSED");
            break;
        case LV_EVENT_CLICKED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_CLICKED");
            break;
        case LV_EVENT_LONG_PRESSED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED");
            break;
        case LV_EVENT_LONG_PRESSED_REPEAT:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED_REPEAT");
            break;
        default:
            break;
    }
}

/**
 * Handle multiple events
 */
void lv_example_event_button(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);

    lv_obj_t * btn_label = lv_label_create(btn);
    lv_label_set_text(btn_label, "Click me!");
    lv_obj_center(btn_label);

    lv_obj_t * info_label = lv_label_create(lv_screen_active());
    lv_label_set_text(info_label, "The last button event:\nNone");

    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_ALL, info_label);
}

#endif

```

### 3.4.4 Evento bubbling

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void event_cb(lv_event_t * e)
{
    /*The original target of the event. Can be the buttons or the container*/
    lv_obj_t * target = lv_event_get_target_obj(e);

    /*The current target is always the container as the event is added to it*/
    lv_obj_t * cont = lv_event_get_current_target_obj(e);

    /*If container was clicked do nothing*/
    if(target == cont) return;

    /*Make the clicked buttons red*/
    lv_obj_set_style_bg_color(target, lv_palette_main(LV_PALETTE_RED), 0);
}

/**
 * Demonstrate event bubbling
 */
void lv_example_event_bubble(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 290, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 30; i++) {
        lv_obj_t * btn = lv_button_create(cont);
        lv_obj_set_size(btn, 70, 50);
        lv_obj_add_flag(btn, LV_OBJ_FLAG_EVENT_BUBBLE);

        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_obj_add_event_cb(cont, event_cb, LV_EVENT_CLICKED, NULL);
}

```

(continues on next page)

(continua dalla pagina precedente)

#endif

### 3.4.5 Event trickle-down

```
#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES || !LV_USE_FLEX
    /* Demonstrate event trickle */
    void lv_example_event_trickle(void)
    {
        lv_obj_t * cont = lv_obj_create(lv_screen_active());
        lv_obj_set_size(cont, 290, 200);
        lv_obj_center(cont);
        lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

        static lv_style_t style_black;
        lv_style_init(&style_black);
        lv_style_set_text_color(&style_black, lv_color_white());
        lv_style_set_bg_color(&style_black, lv_color_black());

        /*Enable event trickle-down on the container*/
        lv_obj_add_flag(cont, LV_OBj_FLAG_EVENT_TRICKLE);

        lv_obj_add_style(cont, &style_black, LV_STATE_PRESSED);

        uint32_t i;
        for(i = 0; i < 9; i++) {
            lv_obj_t * subcont = lv_obj_create(cont);
            lv_obj_set_size(subcont, 70, 50);
            lv_obj_t * label = lv_label_create(subcont);
            lv_label_set_text_fmt(label, "%" LV_PRIu32, i);

            /*Add style to the label when clicked*/
            lv_obj_add_style(subcont, &style_black, LV_STATE_FOCUSED);
        }
    }
#endif
```

### 3.4.6 Evento Draw

```
#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES

static int32_t size = 0;
static bool size_dec = false;

static void timer_cb(lv_timer_t * timer)
{
    lv_obj_invalidate((lv_obj_t *) lv_timer_get_user_data(timer));
    if(size_dec) size--;
    else size++;

    if(size == 50) size_dec = true;
    else if(size == 0) size_dec = false;
}

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *) lv_draw_task_get_draw_dsc(draw_task);
    if(base_dsc->part == LV_PART_MAIN) {
        lv_draw_rect_dsc_t draw_dsc;
        lv_draw_rect_dsc_init(&draw_dsc);
        draw_dsc.bg_color = lv_color_hex(0xffffaa);
        draw_dsc.radius = LV_RADIUS_CIRCLE;
        draw_dsc.border_color = lv_color_hex(0xff5555);
        draw_dsc.border_width = 2;
        draw_dsc.outline_color = lv_color_hex(0xff0000);
        draw_dsc.outline_pad = 3;
        draw_dsc.outline_width = 2;

        lv_area_t a;
        a.x1 = 0;
        a.y1 = 0;
        a.x2 = size;
        a.y2 = size;
        lv_area_t obj_coords;
        lv_obj_get_coords(obj, &obj_coords);
        lv_area_align(&obj_coords, &a, LV_ALIGN_CENTER, 0, 0);

        lv_draw_rect(base_dsc->layer, &draw_dsc, &a);
    }
}

/* Demonstrate the usage of draw event
 */
void lv_example_event_draw(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 200, 200);
    lv_obj_center(cont);
    lv_obj_add_event_cb(cont, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(cont, LV_OBj_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_timer_create(timer_cb, 30, cont);
}
```

## 3.5 Layout

### 3.5.1 Flex

#### Un semplice layout di riga e colonna con flexbox

```
#include "../../../lv_examples.h"
#ifndef LV_USE_FLEX || !LV_BUILD_EXAMPLES

/***
 * A simple row and a column layout with flexbox
 */
void lv_example_flex_1(void)
{
    /*Create a container with ROW flex direction*/
    lv_obj_t * cont_row = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_row, 300, 75);
    lv_obj_align(cont_row, LV_ALIGN_TOP_MID, 0, 5);
    lv_obj_set_flex_flow(cont_row, LV_FLEX_FLOW_ROW);

    /*Create a container with COLUMN flex direction*/
    lv_obj_t * cont_col = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_col, 200, 150);
    lv_obj_align_to(cont_col, cont_row, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
    lv_obj_set_flex_flow(cont_col, LV_FLEX_FLOW_COLUMN);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * obj;
        lv_obj_t * label;

        /*Add items to the row*/
        obj = lv_button_create(cont_row);
        lv_obj_set_size(obj, 100, LV_PCT(100));

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %" LV_PRIu32 "", i);
        lv_obj_center(label);

        /*Add items to the column*/
        obj = lv_button_create(cont_col);
        lv_obj_set_size(obj, LV_PCT(100), LV_SIZE_CONTENT);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %" LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

#### Disporre gli elementi in righe con avvolgimento e spaziatura uniforme

```
#include "../../../lv_examples.h"
#ifndef LV_USE_FLEX || !LV_BUILD_EXAMPLES

/***
 * Arrange items in rows with wrap and place the items to get even space around them.
 */
void lv_example_flex_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_flex_flow(&style, LV_FLEX_FLOW_ROW_WRAP);
    lv_style_set_flex_main_place(&style, LV_FLEX_ALIGN_SPACE_EVENLY);
    lv_style_set_layout(&style, LV_LAYOUT_FLEX);

    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_add_style(cont, &style, 0);

    uint32_t i;
    for(i = 0; i < 8; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);
        lv_obj_add_flag(obj, LV_OBJ_FLAG_CHECKABLE);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

#### Dimostrare la flex grow.

```
#include "../../../lv_examples.h"
#ifndef LV_USE_FLEX || !LV_BUILD_EXAMPLES

/***
 * Demonstrate flex grow.
 */
void lv_example_flex_3(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * obj;
obj = lv_obj_create(cont);
lv_obj_set_size(obj, 40, 40); /*Fix size*/

obj = lv_obj_create(cont);
lv_obj_set_height(obj, 40);
lv_obj_set_flex_grow(obj, 1); /*1 portion from the free space*/

obj = lv_obj_create(cont);
lv_obj_set_height(obj, 40);
lv_obj_set_flex_grow(obj, 2); /*2 portion from the free space*/

obj = lv_obj_create(cont);
lv_obj_set_size(obj, 40, 40); /*Fix size. It is flushed to the right by the "grow" items*/
}

#endif

```

## Dimostrare la flex grow.

```

#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * Reverse the order of flex items
 */
void lv_example_flex_4(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN_REVERSE);

    uint32_t i;
    for(i = 0; i < 6; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 100, 50);

        lv_label_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %" LV_PRIu32, i);
        lv_obj_center(label);
    }
}
#endif

```

## Dimostrare le proprietà di stile degli spazi tra colonne e righe

```

#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row((lv_obj_t *) obj, v, 0);
}

static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column((lv_obj_t *) obj, v, 0);
}

/***
 * Demonstrate the effect of column and row gap style properties
 */
void lv_example_flex_5(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 9; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_label_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, cont);
    lv_anim_set_values(&a, 0, 10);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

    lv_anim_set_exec_cb(&a, row_gap_anim);
    lv_anim_set_duration(&a, 500);
    lv_anim_set_reverse_duration(&a, 500);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, column_gap_anim);
    lv_anim_set_duration(&a, 3000);
    lv_anim_set_reverse_duration(&a, 3000);
    lv_anim_start(&a);
}

#endif

```

## La direzione di base RTL modifica l'ordine degli elementi

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * RTL base direction changes order of the items.
 * Also demonstrate how horizontal scrolling works with RTL.
 */
void lv_example_flex_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_style_base_dir(cont, LV_BASE_DIRRTL, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }
}
#endif
```

## 3.5.2 Grid

### Una semplice grid

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * A simple grid
 */
void lv_example_grid_1(void)
{
    static int32_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_style_grid_column_dsc_array(cont, col_dsc, 0);
    lv_obj_set_style_grid_row_dsc_array(cont, row_dsc, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_layout(cont, LV_LAYOUT_GRID);

    lv_obj_t * label;
    lv_obj_t * obj;

    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_button_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "c%d, r%d", col, row);
        lv_obj_center(label);
    }
}
#endif
```

### Posizionamento e ampiezza delle celle

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * Demonstrate cell placement and span
 */
void lv_example_grid_2(void)
{
    static int32_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);

    lv_obj_t * label;
    lv_obj_t * obj;

    /*Cell to 0:0 and align to the start (left/top) horizontally and vertically too*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 0, 1,
                         LV_GRID_ALIGN_START, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c0, r0");
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Cell to 1;0 and align to the start (left) horizontally and center vertically too*/
obj = lv_obj_create(cont);
lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 1, 1,
                     LV_GRID_ALIGN_CENTER, 0, 1);
label = lv_label_create(obj);
lv_label_set_text(label, "c1, r0");

/*Cell to 2;0 and align to the start (left) horizontally and end (bottom) vertically too*/
obj = lv_obj_create(cont);
lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 2, 1,
                     LV_GRID_ALIGN_END, 0, 1);
label = lv_label_create(obj);
lv_label_set_text(label, "c2, r0");

/*Cell to 1;1 but 2 column wide (span = 2).Set width and height to stretched.*/
obj = lv_obj_create(cont);
lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 1, 2,
                     LV_GRID_ALIGN_STRETCH, 1, 1);
label = lv_label_create(obj);
lv_label_set_text(label, "c1-2, r1");

/*Cell to 0;1 but 2 rows tall (span = 2).Set width and height to stretched.*/
obj = lv_obj_create(cont);
lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 0, 1,
                     LV_GRID_ALIGN_STRETCH, 1, 2);
label = lv_label_create(obj);
lv_label_set_text(label, "c0\nr1-2");
}

#endif

```

## La -free unit- della grid

```

#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate grid's "free unit"
 */
void lv_example_grid_3(void)
{
    /*Column 1: fix width 60 px
     *Column 2: 1 unit from the remaining free space
     *Column 3: 2 unit from the remaining free space*/
    static int32_t col_dsc[] = {60, LV_GRID_FR(1), LV_GRID_FR(2), LV_GRID_TEMPLATE_LAST};

    /*Row 1: fix width 50 px
     *Row 2: 1 unit from the remaining free space
     *Row 3: fix width 50 px*/
    static int32_t row_dsc[] = {50, LV_GRID_FR(1), 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                            LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
}

#endif

```

## Posizionamento delle tracce

```

#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate track placement
 */
void lv_example_grid_4(void)
{
    static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Add space between the columns and move the rows to the bottom (end)*/

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_grid_align(cont, LV_GRID_ALIGN_SPACE_BETWEEN, LV_GRID_ALIGN_END);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * label;
lv_obj_t * obj;
uint8_t i;
for(i = 0; i < 9; i++) {
    uint8_t col = i % 3;
    uint8_t row = i / 3;

    obj = lv_obj_create(cont);
    /*Stretch the cell horizontally and vertically too
     *Set span to 1 to make the cell 1 column/row sized*/
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                         LV_GRID_ALIGN_STRETCH, row, 1);

    label = lv_label_create(obj);
    lv_label_set_text_fmt(label, "%d,%d", col, row);
    lv_obj_center(label);
}
}

#endif

```

## Spazi tra colonne e righe

```

#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row((lv_obj_t *) obj, v, 0);
}

static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column((lv_obj_t *) obj, v, 0);
}

/***
 * Demonstrate column and row gap
 */
void lv_example_grid_5(void)
{
    /*60x60 cells*/
    static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                            LV_GRID_ALIGN_STRETCH, row, 1);
        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, cont);
    lv_anim_set_values(&a, 0, 10);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

    lv_anim_set_exec_cb(&a, row_gap_anim);
    lv_anim_set_duration(&a, 500);
    lv_anim_set_reverse_duration(&a, 500);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, column_gap_anim);
    lv_anim_set_duration(&a, 3000);
    lv_anim_set_reverse_duration(&a, 3000);
    lv_anim_start(&a);
}

#endif

```

## Direzione RTL sulla griglia

```

#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * Demonstrate RTL direction on grid
 */
void lv_example_grid_6(void)
{
    static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_center(cont);
lv_obj_set_style_base_dir(cont, LV_BASE_DIRRTL, 0);
lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

lv_obj_t * label;
lv_obj_t * obj;
uint8_t i;
for(i = 0; i < 9; i++) {
    uint8_t col = i % 3;
    uint8_t row = i / 3;

    obj = lv_obj_create(cont);
    /*Stretch the cell horizontally and vertically too
     *Set span to 1 to make the cell 1 column/row sized*/
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                         LV_GRID_ALIGN_STRETCH, row, 1);

    label = lv_label_create(obj);
    lv_label_set_text_fmt(label, "%d,%d", col, row);
    lv_obj_center(label);
}

#endif
}

```

## 3.6 Scrolling

### 3.6.1 Scrolling nidificato

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

static lv_obj_t * panel;
static lv_obj_t * save_button;
static lv_obj_t * restore_button;
static int saved_scroll_x;
static int saved_scroll_y;

static void scroll_update_cb(lv_event_t * e);
static void button_event_cb(lv_event_t * e);

static void scroll_update_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    LV_LOG("scroll info: x:%3" LV_PRIId32 ", y:%3" LV_PRIId32 ", top:%3" LV_PRIId32 ", "
          "bottom:%3" LV_PRIId32 ", left:%3" LV_PRIId32 ", right:%3" LV_PRIId32 "\n",
          lv_obj_get_scroll_x(panel),
          lv_obj_get_scroll_y(panel),
          lv_obj_get_scroll_top(panel),
          lv_obj_get_scroll_bottom(panel),
          lv_obj_get_scroll_left(panel),
          lv_obj_get_scroll_right(panel)
    );
}

static void button_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    if(obj == save_button) {
        saved_scroll_x = lv_obj_get_scroll_x(panel);
        saved_scroll_y = lv_obj_get_scroll_y(panel);
    }
    else {
        lv_obj_scroll_to(panel, saved_scroll_x, saved_scroll_y, LV_ANIM_ON);
    }
}

/***
 * Demonstrate how scrolling appears automatically
 */
void lv_example_scroll_1(void)
{
    /*Create an object with the new style*/
    lv_obj_t * scr;
    scr = lv_screen_active();
    panel = lv_obj_create(scr);
    lv_obj_set_size(panel, 200, 200);
    lv_obj_align(panel, LV_ALIGN_CENTER, 44, 0);

    lv_obj_t * child;
    lv_obj_t * label;

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 0, 0);
    lv_obj_set_size(child, 70, 70);
    label = lv_label_create(child);
    lv_label_set_text(label, "Zero");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 160, 80);
    lv_obj_set_size(child, 80, 80);

    lv_obj_t * child2 = lv_button_create(child);
    lv_obj_set_size(child2, 100, 50);

    label = lv_label_create(child2);
    lv_label_set_text(label, "Right");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 40, 160);
    lv_obj_set_size(child, 100, 70);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

label = lv_label_create(child);
lv_label_set_text(label, "Bottom");
lv_obj_center(label);

/* When LV_OBJ_FLAG_SCROLL_ELASTIC is cleared, scrolling does not go past edge boundaries. */
/* lv_obj_clear_flag(panel, LV_OBJ_FLAG_SCROLL_ELASTIC); */

/* Call `scroll_update_cb` while panel is being scrolled. */
lv_obj_add_event_cb(panel, scroll_update_cb, LV_EVENT_SCROLL, NULL);

/* Set up buttons that save and restore scroll position. */
save_button = lv_button_create(scr);
restore_button = lv_button_create(scr);
lv_obj_t * lbl;
lbl = lv_label_create(save_button);
lv_label_set_text_static(lbl, "Save");
lbl = lv_label_create(restore_button);
lv_label_set_text_static(lbl, "Restore");
lv_obj_align_to(save_button, panel, LV_ALIGN_OUT_LEFT_MID, -10, -20);
lv_obj_align_to(restore_button, panel, LV_ALIGN_OUT_LEFT_MID, -10, 20);
lv_obj_add_event_cb(save_button, button_event_cb, LV_EVENT_CLICKED, NULL);
lv_obj_add_event_cb(restore_button, button_event_cb, LV_EVENT_CLICKED, NULL);
}

#endif

```

## 3.6.2 Aggancio

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void sw_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * sw = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_t * list = (lv_obj_t *) lv_event_get_user_data(e);

        if(lv_obj_has_state(sw, LV_STATE_CHECKED)) lv_obj_add_flag(list, LV_OBJ_FLAG_SCROLL_ONE);
        else lv_obj_remove_flag(list, LV_OBJ_FLAG_SCROLL_ONE);
    }
}

/**
 * Show an example to scroll snap
 */
void lv_example_scroll_2(void)
{
    lv_obj_t * panel = lv_obj_create(lv_screen_active());
    lv_obj_set_size(panel, 280, 120);
    lv_obj_set_scroll_snap_x(panel, LV_SCROLL_SNAP_CENTER);
    lv_obj_set_flex_flow(panel, LV_FLEX_FLOW_ROW);
    lv_obj_align(panel, LV_ALIGN_CENTER, 0, 20);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * btn = lv_button_create(panel);
        lv_obj_set_size(btn, 150, lv_pct(100));

        lv_obj_t * label = lv_label_create(btn);
        if(i == 3) {
            lv_label_set_text_fmt(label, "Panel %" LV_PRIu32"\nno snap", i);
            lv_obj_remove_flag(btn, LV_OBJ_FLAG_SNAPPABLE);
        } else {
            lv_label_set_text_fmt(label, "Panel %" LV_PRIu32, i);
        }

        lv_obj_center(label);
    }
    lv_obj_update_snap(panel, LV_ANIM_ON);
}

#endif

```

## 3.6.3 Pulsante mobile

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST

static uint32_t btn_cnt = 1;

static void float_button_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * float_btn = lv_event_get_target_obj(e);

    if(code == LV_EVENT_CLICKED) {
        lv_obj_t * list = (lv_obj_t *) lv_event_get_user_data(e);
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_obj_t * list_btn = lv_list_add_button(list, LV_SYMBOL_AUDIO, buf);
        btn_cnt++;
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

        lv_obj_move_foreground(float_btn);
        lv_obj_scroll_to_view(list_btn, LV_ANIM_ON);
    }

    /**
     * Create a list with a floating button
     */
void lv_example_scroll_3(void)
{
    lv_obj_t * list = lv_list_create(lv_screen_active());
    lv_obj_set_size(list, 280, 220);
    lv_obj_center(list);

    for(btn_cnt = 1; btn_cnt <= 2; btn_cnt++) {
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_list_add_button(list, LV_SYMBOL_AUDIO, buf);
    }

    lv_obj_t * float_btn = lv_button_create(list);
    lv_obj_set_size(float_btn, 50, 50);
    lv_obj_add_flag(float_btn, LV_OBJ_FLAG_FLOATING);
    lv_obj_align(float_btn, LV_ALIGN_BOTTOM_RIGHT, 0, -lv_obj_get_style_pad_right(list, LV_PART_MAIN));
    lv_obj_add_event_cb(float_btn, float_button_event_cb, LV_EVENT_ALL, list);
    lv_obj_set_style_radius(float_btn, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_image_src(float_btn, LV_SYMBOL_PLUS, 0);
    lv_obj_set_style_text_font(float_btn, lv_theme_get_font_large(float_btn), 0);
}

#endif

```

### 3.6.4 Stile delle barre di scorrimento

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST

    /**
     * Styling the scrollbars
     */
void lv_example_scroll_4(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_set_size(obj, 200, 100);
    lv_obj_center(obj);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label,
        "Lorem ipsum dolor sit amet, consectetur adipiscing elit.\n"
        "Etiam dictum, tortor vestibulum lacinia laoreet, mi neque consectetur neque, vel mattis odio dolor egestas ligula. \n"
        "Sed vestibulum sapien nulla, id convallis ex porttitor nec. \n"
        "Duis et massa eu libero accumsan faucibus a in arcu. \n"
        "Ut pulvinar odio lorem, vel tempus turpis condimentum quis. Nam consectetur condimentum sem in auctor. \n"
        "Sed nisl augue, venenatis in blandit et, gravida ac tortor. \n"
        "Etiam dapibus elementum suscipit. \n"
        "Proin mollis sollicitudin convallis. \n"
        "Integer dapibus tempus arcu nec viverra. \n"
        "Donec molestie nulla enim, eu interdum velit placerat quis. \n"
        "Donec id efficitur risus, at molestie turpis. \n"
        "Suspendisse vestibulum consectetur nunc ut commodo. \n"
        "Fusce molestie rhoncus nisi sit amet tincidunt. \n"
        "Suspendisse a nunc ut magna ornare volutpat.");
}

/*Remove the style of scrollbar to have clean start*/
lv_obj_remove_style(obj, NULL, LV_PART_SCROLLBAR | LV_STATE_ANY);

/*Create a transition the animate the some properties on state change*/
static const lv_style_prop_t props[] = {LV_STYLE_BG_OPA, LV_STYLE_WIDTH, 0};
static lv_style_transition_dsc_t trans;
lv_style_transition_dsc_init(&trans, props, lv_anim_path_linear, 200, 0, NULL);

/*Create a style for the scrollbars*/
static lv_style_t style;
lv_style_init(&style);
lv_style_set_width(&style, 4);      /*Width of the scrollbar*/
lv_style_set_length(&style, 20);    /*Length of the scrollbar*/
lv_style_set_pad_right(&style, 5);   /*Space from the parallel side*/
lv_style_set_pad_top(&style, 5);    /*Space from the perpendicular side*/

lv_style_set_radius(&style, 2);
lv_style_set_bg_opa(&style, LV_OPA_70);
lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
lv_style_set_border_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 3));
lv_style_set_border_width(&style, 2);
lv_style_set_shadow_width(&style, 8);
lv_style_set_shadow_spread(&style, 2);
lv_style_set_shadow_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 1));

lv_style_set_transition(&style, &trans);

/*Make the scrollbars wider and use 100% opacity when scrolled*/
static lv_style_t style_scrolled;
lv_style_init(&style_scrolled);
lv_style_set_width(&style_scrolled, 8);
lv_style_set_bg_opa(&style_scrolled, LV_OPA_COVER);

lv_obj_add_style(obj, &style, LV_PART_SCROLLBAR);
lv_obj_add_style(obj, &style_scrolled, LV_PART_SCROLLBAR | LV_STATE_SCROLLED);
}

#endif

```

### 3.6.5 Scorrimento da destra a sinistra

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_FONT_DEJAVU_16_PERSIAN_HEBREW

/***
 * Scrolling with Right To Left base direction
 */
void lv_example_scroll_5(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_set_style_base_dir(obj, LV_BASE_DIRRTL, 0);
    lv_obj_set_size(obj, 200, 100);
    lv_obj_center(obj);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label,
        "من دن اوح طوق مطقاچ و (RAM) یقدامهت سپرسد مطقاچ ایاراد هک تساوا هدنزا در بزرگی ایاهنگ (Microcontroller) رلورون کیویلگن اه رلورون کیویلگن اه");

    lv_obj_set_width(label, 400);
    lv_obj_set_style_text_font(label, &lv_font_dejavu_16_persian_hebrew, 0);
}

#endif
```

### 3.6.6 Traslazione durante lo scorrimento

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void scroll_event_cb(lv_event_t * e)
{
    lv_obj_t * cont = lv_event_get_target_obj(e);

    lv_area_t cont_a;
    lv_obj_get_coords(cont, &cont_a);
    int32_t cont_y_center = cont_a.y1 + lv_area_get_height(&cont_a) / 2;

    int32_t r = lv_obj_get_height(cont) * 7 / 10;
    int32_t i;
    int32_t child_cnt = (int32_t)lv_obj_get_child_count(cont);
    for(i = 0; i < child_cnt; i++) {
        lv_obj_t * child = lv_obj_get_child(cont, i);
        lv_area_t child_a;
        lv_obj_get_coords(child, &child_a);

        int32_t child_y_center = child_a.y1 + lv_area_get_height(&child_a) / 2;

        int32_t diff_y = child_y_center - cont_y_center;
        diff_y = LV_ABS(diff_y);

        /*Get the x of diff_y on a circle.*/
        int32_t x;
        /*If diff_y is out of the circle use the last point of the circle (the radius)*/
        if(diff_y >= r) {
            x = r;
        } else {
            /*Use Pythagoras theorem to get x from radius and y*/
            uint32_t x_sqr = r * r - diff_y * diff_y;
            lv_sqrt_res_t res;
            lv_sqrt(x_sqr, &res, 0x8000); /*Use lvgl's built in sqrt root function*/
            x = r - res.i;
        }

        /*Translate the item by the calculated X coordinate*/
        lv_obj_set_style_translate_x(child, x, 0);

        /*Use some opacity with larger translations*/
        lv_opa_t opa = (lv_opa_t)lv_map(x, 0, r, LV_OPA_TRANSPI, LV_OPA_COVER);
        lv_obj_set_style_opa(child, LV_OPA_COVER - opa, 0);
    }
}

/***
 * Translate the object as they scroll
 */
void lv_example_scroll_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 200, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN);
    lv_obj_add_event_cb(cont, scroll_event_cb, LV_EVENT_SCROLL, NULL);
    lv_obj_set_style_radius(cont, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(cont, true, 0);
    lv_obj_set_scroll_dir(cont, LV_DIR_VER);
    lv_obj_set_scroll_snap_y(cont, LV_SCROLL_SNAP_CENTER);
    lv_obj_set_scrollbar_mode(cont, LV_SCROLLBAR_MODE_OFF);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * btn = lv_button_create(cont);
        lv_obj_set_width(btn, lv_pct(100));

        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "Button %" LV_PRIu32, i);
    }

    /*Update the buttons position manually for first*/
    lv_obj_send_event(cont, LV_EVENT_SCROLL, NULL);

    /*Be sure the fist button is in the middle*/
    lv_obj_scroll_to_view(lv_obj_get_child(cont, 0), LV_ANIM_OFF);
}
```

(continues on next page)

(continua dalla pagina precedente)

#endif

### 3.6.7 Scrolling infinito

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static lv_obj_t * high_label;
static lv_obj_t * low_label;
static int32_t top_num;
static int32_t bottom_num;
static bool update_scroll_running = false;

static lv_obj_t * load_item(lv_obj_t * parent, int32_t num)
{
    lv_obj_t * obj = (lv_obj_t *) lv_obj_create(parent);
    lv_obj_set_size(obj, LV_PCT(100), LV_SIZE_CONTENT);
    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text_fmt(label, "%" PRId32, num);
    return obj;
}

static void update_scroll(lv_obj_t * obj)
{
    /* do not re-enter this function when `lv_obj_scroll_by` triggers this callback again. */
    if(update_scroll_running) return;
    update_scroll_running = true;

    int32_t top_num_original = top_num;
    int32_t bottom_num_original = bottom_num;

    /* load items we're getting close to */
    while(bottom_num > -30 && lv_obj_get_scroll_bottom(obj) < 200) {
        bottom_num -= 1;
        load_item(obj, bottom_num);
        lv_obj_update_layout(obj);
        LV_LOG_USER("loaded bottom num: %" PRId32, bottom_num);
    }
    while(top_num < 30 && lv_obj_get_scroll_top(obj) < 200) {
        top_num += 1;
        int32_t bottom_before = lv_obj_get_scroll_bottom(obj);
        lv_obj_t * new_item = load_item(obj, top_num);
        lv_obj_move_to_index(new_item, 0);
        lv_obj_update_layout(obj);
        int32_t bottom_after = lv_obj_get_scroll_bottom(obj);
        lv_obj_scroll_by(obj, 0, bottom_before - bottom_after, LV_ANIM_OFF);
        LV_LOG_USER("loaded top num: %" PRId32, top_num);
    }

    /* delete far-away items */
    while(lv_obj_get_scroll_bottom(obj) > 600) {
        bottom_num += 1;
        lv_obj_t * child = lv_obj_get_child(obj, -1);
        lv_obj_delete(child);
        lv_obj_update_layout(obj);
        LV_LOG_USER("deleted bottom num: %" PRId32, bottom_num);
    }
    while(lv_obj_get_scroll_top(obj) > 600) {
        top_num -= 1;
        int32_t bottom_before = lv_obj_get_scroll_bottom(obj);
        lv_obj_t * child = lv_obj_get_child(obj, 0);
        lv_obj_delete(child);
        lv_obj_update_layout(obj);
        int32_t bottom_after = lv_obj_get_scroll_bottom(obj);
        lv_obj_scroll_by(obj, 0, bottom_before - bottom_after, LV_ANIM_OFF);
        LV_LOG_USER("deleted top num: %" PRId32, top_num);
    }

    if(top_num != top_num_original) {
        lv_label_set_text_fmt(high_label, "current largest\nloaded value:\n%" PRId32, top_num);
    }
    if(bottom_num != bottom_num_original) {
        lv_label_set_text_fmt(low_label, "current smallest\nloaded value:\n%" PRId32, bottom_num);
    }
    update_scroll_running = false;
}

static void scroll_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    update_scroll(obj);
}

static void checkbox_cb(lv_event_t * e)
{
    lv_obj_t * checkbox = lv_event_get_target_obj(e);
    lv_obj_t * obj = (lv_obj_t *) lv_event_get_user_data(e);
    bool checked = lv_obj_has_state(checkbox, LV_STATE_CHECKED);
    lv_obj_set_style_opa(obj, checked ? LV_OPA_COVER : LV_OPA_TRANSP, LV_PART_SCROLLBAR);
}

/**
 * Dynamically load widgets while scrolling
 */
void lv_example_scroll_7(void)
{
    lv_obj_t * scr = lv_screen_active();
    lv_obj_t * obj = lv_obj_create(scr);
    lv_obj_set_size(obj, 160, 220);
    lv_obj_align(obj, LV_ALIGN_RIGHT_MID, -10, 0);
    lv_obj_set_flex_flow(obj, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_style_opa(obj, LV_OPA_TRANSP, LV_PART_SCROLLBAR);
    high_label = lv_label_create(scr);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_label_set_text_static(high_label, "current largest\nloaded value:");
lv_obj_align(high_label, LV_ALIGN_TOP_LEFT, 10, 10);

lv_obj_t * checkbox = lv_checkbox_create(scr);
lv_checkbox_set_text_static(checkbox, "show\nscrollbar");
lv_obj_align(checkbox, LV_ALIGN_LEFT_MID, 10, 0);
lv_obj_add_event_cb(checkbox, checkbox_cb, LV_EVENT_VALUE_CHANGED, obj);

low_label = lv_label_create(scr);
lv_label_set_text_static(low_label, "current smallest\nloaded value:");
lv_obj_align(low_label, LV_ALIGN_BOTTOM_LEFT, 10, -10);

load_item(obj, 3);
/* These counters hold the highest/lowest number currently loaded. */
top_num = 3;
bottom_num = 3;

lv_obj_update_layout(obj);
update_scroll(obj);
lv_obj_add_event_cb(obj, scroll_cb, LV_EVENT_SCROLL, NULL);
}

#endif

```

### 3.6.8 Scrolling circolare

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

/*
 * Circular list implementation based on:
 * Reference: https://blog.csdn.net/TQW4321/article/details/145434819
 */

/* Applies to both width (for row items) and height (for column items) */
#define ITEM_SIZE     80

/* Function to calculate the total content width of the container */
static int32_t get_content_width(lv_obj_t * cont)
{
    int32_t w = 0;
    int32_t pad_column = lv_obj_get_style_pad_column(cont, LV_PART_MAIN);
    int32_t total = (int32_t)lv_obj_get_child_count(cont);

    for(int32_t i = 0; i < total; i++) {
        w += lv_obj_get_width(lv_obj_get_child(cont, i));
        if(i < total - 1) w += pad_column;
    }
    return w + lv_obj_get_style_pad_left(cont, LV_PART_MAIN)
        + lv_obj_get_style_pad_right(cont, LV_PART_MAIN);
}

/* Scroll event callback for row layout */
static void cont_row_scroll_event_cb(lv_event_t * e)
{
    static bool is_adjusting = false;
    lv_obj_t * cont = lv_event_get_current_target_obj(e);

    if(!is_adjusting) {
        is_adjusting = true;
        int32_t scroll_x = lv_obj_get_scroll_x(cont);
        int32_t cont_w = lv_obj_get_width(cont);
        int32_t content_w = get_content_width(cont);

        /* Use ITEM_SIZE as horizontal item width */
        const int32_t item_width = ITEM_SIZE;

        if(scroll_x < 0) {
            lv_obj_t * last_child = lv_obj_get_child(cont, (int32_t)(lv_obj_get_child_count(cont) - 1));
            lv_obj_move_to_index(last_child, 0);
            lv_obj_scroll_to_x(cont, scroll_x + item_width, LV_ANIM_OFF);
        } else if(scroll_x > content_w - cont_w) {
            lv_obj_t * first_child = lv_obj_get_child(cont, 0);
            lv_obj_move_to_index(first_child, (int32_t)(lv_obj_get_child_count(cont) - 1));
            lv_obj_scroll_to_x(cont, scroll_x - item_width, LV_ANIM_OFF);
        }
        is_adjusting = false;
    }
}

/* Function to calculate the total content height of the container */
static int32_t get_content_height(lv_obj_t * cont)
{
    int32_t h = 0;
    int32_t pad_row = lv_obj_get_style_pad_row(cont, LV_PART_MAIN);
    int32_t total = (int32_t)lv_obj_get_child_count(cont);

    for(int i = 0; i < total; i++) {
        h += lv_obj_get_height(lv_obj_get_child(cont, i));
        if(i < total - 1) h += pad_row;
    }
    return h + lv_obj_get_style_pad_top(cont, LV_PART_MAIN)
        + lv_obj_get_style_pad_bottom(cont, LV_PART_MAIN);
}

/* Scroll event callback for column layout */
static void cont_col_scroll_event_cb(lv_event_t * e)
{
    static bool is_adjusting = false;
    lv_obj_t * cont = lv_event_get_current_target_obj(e);

    if(!is_adjusting) {
        is_adjusting = true;
        int32_t scroll_y = lv_obj_get_scroll_y(cont);
        int32_t cont_h = lv_obj_get_height(cont);
        int32_t content_h = get_content_height(cont);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/* Use ITEM_SIZE as vertical item height */
const int32_t item_height = ITEM_SIZE;

if(scroll_y <= 0) {
    lv_obj_t * last_child = lv_obj_get_child(cont, (int32_t)(lv_obj_get_child_count(cont) - 1));
    lv_obj_move_to_index(last_child, 0);
    lv_obj_scroll_to_y(cont, scroll_y + item_height, LV_ANIM_OFF);
}
else if(scroll_y > content_h - cont_h) {
    lv_obj_t * first_child = lv_obj_get_child(cont, 0);
    lv_obj_move_to_index(first_child, (int32_t)(lv_obj_get_child_count(cont) - 1));
    lv_obj_scroll_to_y(cont, scroll_y - item_height, LV_ANIM_OFF);
}
is_adjusting = false;
}

void lv_example_scroll_8(void)
{
    /* Create a scroll container with ROW flex direction */
    lv_obj_t * cont_row = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_row, 300, 75);
    lv_obj_align(cont_row, LV_ALIGN_TOP_MID, 0, 5);
    lv_obj_set_flex_flow(cont_row, LV_FLEX_FLOW_ROW);
    lv_obj_add_event_cb(cont_row, cont_row_scroll_event_cb, LV_EVENT_SCROLL, NULL);
    /* Hide scrollbar visuals */
    lv_obj_set_scrollbar_mode(cont_row, LV_SCROLLBAR_MODE_OFF);

    /* Create a scroll container with COLUMN flex direction */
    lv_obj_t * cont_col = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_col, 200, 150);
    lv_obj_align_to(cont_col, cont_row, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
    lv_obj_set_flex_flow(cont_col, LV_FLEX_FLOW_COLUMN);
    lv_obj_add_event_cb(cont_col, cont_col_scroll_event_cb, LV_EVENT_SCROLL, NULL);
    /* Hide scrollbar visuals */
    lv_obj_set_scrollbar_mode(cont_col, LV_SCROLLBAR_MODE_OFF);

    /* If the number of items is less than 3, scrolling may not be needed or may cause unexpected behavior. */
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * obj;
        lv_obj_t * label;

        /*Add items to the row*/
        obj = lv_button_create(cont_row);
        lv_obj_set_size(obj, ITEM_SIZE, LV_PCT(100));

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item %" LV_PRIu32, i + 1);
        lv_obj_center(label);

        /*Add items to the column*/
        obj = lv_button_create(cont_col);
        lv_obj_set_size(obj, LV_PCT(100), ITEM_SIZE);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item %" LV_PRIu32, i + 1);
        lv_obj_center(label);
    }
}

#endif

```

## 3.7 I Widget

### 3.7.1 Widget Base

#### Oggetti base con stili personalizzati

```

#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

void lv_example_obj_1(void)
{
    lv_obj_t * obj1;
    obj1 = lv_obj_create(lv_screen_active());
    lv_obj_set_size(obj1, 100, 50);
    lv_obj_align(obj1, LV_ALIGN_CENTER, -60, -30);

    static lv_style_t style_shadow;
    lv_style_init(&style_shadow);
    lv_style_set_shadow_width(&style_shadow, 10);
    lv_style_set_shadow_spread(&style_shadow, 5);
    lv_style_set_shadow_color(&style_shadow, lv_palette_main(LV_PALETTE_BLUE));

    lv_obj_t * obj2;
    obj2 = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj2, &style_shadow, 0);
    lv_obj_align(obj2, LV_ALIGN_CENTER, 60, 30);
}
#endif

```

## Creare un oggetto trascinabile

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

static void drag_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    lv_indev_t * indev = lv_indev_active();
    if(indev == NULL) return;

    lv_point_t vect;
    lv_indev_get_vect(indev, &vect);

    int32_t x = lv_obj_get_x_aligned(obj) + vect.x;
    int32_t y = lv_obj_get_y_aligned(obj) + vect.y;
    lv_obj_set_pos(obj, x, y);
}

/** 
 * Make an object draggable.
 */
void lv_example_obj_2(void)
{
    lv_obj_t * obj;
    obj = lv_obj_create(lv_screen_active());
    lv_obj_set_size(obj, 150, 100);
    lv_obj_add_event_cb(obj, drag_event_handler, LV_EVENT_PRESSING, NULL);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label, "Drag me");
    lv_obj_center(label);
}

#endif
```

## Trasformare un oggetto usando una matrice 3x3

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_DRAW_TRANSFORM_USE_MATRIX

static void timer_cb(lv_timer_t * timer)
{
    lv_obj_t * obj = (lv_obj_t *) lv_timer_get_user_data(timer);

    static float value = 0.1f;
    lv_matrix_t matrix;
    lv_matrix_identity(&matrix);
    lv_matrix_scale(&matrix, value, 1);
    lv_matrix_rotate(&matrix, value * 360);
    lv_obj_set_transform(obj, &matrix);

    value += 0.01f;

    if(value > 2.0f) {
        lv_obj_reset_transform(obj);
        value = 0.1f;
    }
}

void lv_example_obj_3(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_center(obj);

    lv_timer_create(timer_cb, 20, obj);
}

#else

void lv_example_obj_3(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text_static(label, "LV_DRAW_TRANSFORM_USE_MATRIX is not enabled");
    lv_obj_center(label);
}

#endif /*LV_DRAW_TRANSFORM_USE_MATRIX*/
#endif /*LV_BUILD_EXAMPLES*/
```

## 3.7.2 Immagine animata

### Immagine animata semplice

```
#include "../../lv_examples.h"
#if LV_USE_ANIMIMG && LV_BUILD_EXAMPLES
LV_IMAGE_DECLARE(animimg001);
LV_IMAGE_DECLARE(animimg002);
LV_IMAGE_DECLARE(animimg003);

static const lv_image_dsc_t * anim_imgs[3] = {
    &animimg001,
    &animimg002,
    &animimg003,
};

void lv_example_animimg_1(void)
{
    lv_obj_t * animimg0 = lv_animimg_create(lv_screen_active());
    lv_obj_center(animimg0);
    lv_animimg_set_src(animimg0, (const void **) anim_imgs, 3);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_animimg_set_duration(animimg0, 1000);
lv_animimg_set_repeat_count(animimg0, LV_ANIM_REPEAT_INFINITE);
lv_animimg_start(animimg0);
}

#endif

```

### 3.7.3 Arco

#### Arco semplice

```

#include "../../../lv_examples.h"
#if LV_USE_ARC && LV_BUILD_EXAMPLES

static void value_changed_event_cb(lv_event_t * e);

void lv_example_arc_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());

    /*Create an Arc*/
    lv_obj_t * arc = lv_arc_create(lv_screen_active());
    lv_obj_set_size(arc, 150, 150);
    lv_arc_set_rotation(arc, 135);
    lv_arc_set_bg_angles(arc, 0, 270);
    lv_arc_set_value(arc, 10);
    lv_obj_center(arc);
    lv_obj_add_event_cb(arc, value_changed_event_cb, LV_EVENT_VALUE_CHANGED, label);

    /*Manually update the label for the first time*/
    lv_obj_send_event(arc, LV_EVENT_VALUE_CHANGED, NULL);
}

static void value_changed_event_cb(lv_event_t * e)
{
    lv_obj_t * arc = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *)lv_event_get_user_data(e);

    lv_label_set_text_fmt(label, "%" LV_PRIId32 "%%", lv_arc_get_value(arc));

    /*Rotate the label to the current position of the arc*/
    lv_arc_rotate_obj_to_angle(arc, label, 25);
}

#endif

```

#### Loader con Arc

```

#include "../../../lv_examples.h"
#if LV_USE_ARC && LV_BUILD_EXAMPLES

static void set_angle(void * obj, int32_t v)
{
    lv_arc_set_value((lv_obj_t *)obj, v);
}

/**
 * Create an arc which acts as a loader.
 */
void lv_example_arc_2(void)
{
    /*Create an Arc*/
    lv_obj_t * arc = lv_arc_create(lv_screen_active());
    lv_arc_set_rotation(arc, 270);
    lv_arc_set_bg_angles(arc, 0, 360);
    lv_obj_remove_style(arc, NULL, LV_PART_KNOB); /*Be sure the knob is not displayed*/
    lv_obj_remove_flag(arc, LV_OBJ_FLAG_CLICKABLE); /*To not allow adjusting by click*/
    lv_obj_center(arc);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, arc);
    lv_anim_set_exec_cb(&a, set_angle);
    lv_anim_set_duration(&a, 1000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE); /*Just for the demo*/
    lv_anim_set_repeat_delay(&a, 500);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_start(&a);
}

#endif

```

#### Grafico a torta con sezioni cliccabili tramite Archi

```

#include "../../../lv_examples.h"
#if LV_USE_ARC && LV_BUILD_EXAMPLES
#define CHART_SIZE 150
#define SLICE_OFFSET 20

typedef struct {
    int start_angle;
    int end_angle;
    int mid_angle;
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_point_t home;
    bool out;
} slice_info_t;

typedef struct {
    lv_obj_t * obj;
    int start_x;
    int start_y;
    int end_x;
    int end_y;
} slice_anim_data_t;

static float angle_accum = 0.0f;
static slice_info_t * active_info = NULL;
static lv_obj_t * active_arc = NULL;

static void anim_move_cb(void * var, int32_t v)
{
    slice_anim_data_t * d = (slice_anim_data_t *) var;

    int32_t x = d->start_x + ((d->end_x - d->start_x) * v) / 100;
    int32_t y = d->start_y + ((d->end_y - d->start_y) * v) / 100;
    lv_obj_set_pos(d->obj, x, y);
}

static void anim_cleanup_cb(lv_anim_t * a)
{
    lv_free(a->var);
}

static void arc_click_cb(lv_event_t * e)
{
    lv_obj_t * arc = lv_event_get_target_obj(e);
    slice_info_t * info = (slice_info_t *)lv_event_get_user_data(e);

    int32_t x_off = (SLICE_OFFSET * lv_trigo_cos(info->mid_angle)) >> LV_TRIGO_SHIFT;
    int32_t y_off = (SLICE_OFFSET * lv_trigo_sin(info->mid_angle)) >> LV_TRIGO_SHIFT;

    if(active_info && active_info != info && active_info->out) {
        slice_anim_data_t * anim_back = (slice_anim_data_t *) lv_malloc(sizeof(slice_anim_data_t));
        anim_back->obj = active_arc;
        anim_back->start_x = lv_obj_get_x(active_arc) - SLICE_OFFSET;
        anim_back->start_y = lv_obj_get_y(active_arc) - SLICE_OFFSET;
        anim_back->end_x = active_info->home.x;
        anim_back->end_y = active_info->home.y;

        active_info->out = false;

        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, anim_back);
        lv_anim_set_exec_cb(&a, anim_move_cb);
        lv_anim_set_time(&a, 200);
        lv_anim_set_values(&a, 0, 100);
        lv_anim_set_deleted_cb(&a, anim_cleanup_cb);
        lv_anim_start(&a);
    }

    int target_x, target_y;
    if(info->out) {
        target_x = info->home.x;
        target_y = info->home.y;
        info->out = false;
        active_info = NULL;
        active_arc = NULL;
    }
    else {
        target_x = info->home.x + x_off;
        target_y = info->home.y + y_off;
        info->out = true;
        active_info = info;
        active_arc = arc;
    }

    slice_anim_data_t * anim_data = (slice_anim_data_t *) lv_malloc(sizeof(slice_anim_data_t));
    anim_data->obj = arc;
    anim_data->start_x = lv_obj_get_x(arc) - SLICE_OFFSET;
    anim_data->start_y = lv_obj_get_y(arc) - SLICE_OFFSET;
    anim_data->end_x = target_x;
    anim_data->end_y = target_y;

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, anim_data);
    lv_anim_set_exec_cb(&a, anim_move_cb);
    lv_anim_set_time(&a, 200);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_deleted_cb(&a, anim_cleanup_cb);
    lv_anim_start(&a);
}

static void create_slice(lv_obj_t * parent, int percentage, lv_color_t color)
{
    if(percentage <= 0) return;

    float slice_angle = (percentage * 360.0f) / 100.0f;
    int start = (int)(angle_accum + 0.5f);
    angle_accum += slice_angle;
    int end = (int)(angle_accum + 0.5f);
    if(end > 360) end = 360;

    lv_obj_t * arc = lv_arc_create(parent);
    lv_obj_set_size(arc, CHART_SIZE, CHART_SIZE);
    lv_obj_center(arc);

    lv_arc_set_mode(arc, LV_ARC_MODE_NORMAL);
    lv_arc_set_bg_start_angle(arc, start);
    lv_arc_set_bg_end_angle(arc, end);

    lv_obj_set_style_arc_width(arc, CHART_SIZE / 2, LV_PART_MAIN);
    lv_obj_set_style_arc_width(arc, 0, LV_PART_INDICATOR);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_style_arc_color(arc, color, LV_PART_MAIN);
lv_obj_set_style_arc_rounded(arc, false, LV_PART_MAIN);
lv_obj_remove_style(arc, NULL, LV_PART_KNOB);
lv_obj_add_flag(arc, LV_OBJ_FLAG_ADV_HITTEST);

lv_obj_t * label = lv_label_create(arc);
lv_label_set_text_fmt(label, "%d%", percentage);
int mid_angle = start + ((end - start) / 2);
int radius = CHART_SIZE / 4;
int x_offset = (radius * lv_trigo_cos(mid_angle)) >> LV_TRIGO_SHIFT;
int y_offset = (radius * lv_trigo_sin(mid_angle)) >> LV_TRIGO_SHIFT;

lv_obj_align(label, LV_ALIGN_CENTER, x_offset, y_offset);

slice_info_t * info = (slice_info_t *) lv_malloc(sizeof(slice_info_t));
info->start_angle = start;
info->end_angle = end;
info->mid_angle = mid_angle;
info->out = false;
info->home.x = lv_obj_get_x(arc);
info->home.y = lv_obj_get_y(arc);
lv_obj_add_event_cb(arc, arc_click_cb, LV_EVENT_CLICKED, info);
}

void lv_example_arc_3(void)
{
    /* Root container: flex row */
    lv_obj_t * root = lv_obj_create(lv_screen_active());
    lv_obj_set_size(root, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_center(root);
    lv_obj_set_flex_flow(root, LV_FLEX_FLOW_ROW);
    lv_obj_set_flex_align(root, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    lv_obj_set_style_pad_all(root, 0, LV_PART_MAIN);
    lv_obj_set_style_border_width(root, 0, LV_PART_MAIN);
    lv_obj_set_style_border_color(root, lv_palettes_main(LV_PALETTE_RED), LV_PART_MAIN);
    lv_obj_set_style_bg_opa(root, LV_OPA_TRANSP, LV_PART_MAIN);
    lv_obj_remove_flag(root, LV_OBJ_FLAG_SCROLLABLE);

    /* Slices container */
    lv_obj_t * slices_container = lv_obj_create(root);
    lv_obj_set_size(slices_container, CHART_SIZE + 2 * SLICE_OFFSET, CHART_SIZE + 2 * SLICE_OFFSET);
    lv_obj_set_style_pad_all(slices_container, 0, LV_PART_MAIN);
    lv_obj_set_style_margin_all(slices_container, 0, LV_PART_MAIN);
    lv_obj_set_style_border_width(slices_container, 0, LV_PART_MAIN);
    lv_obj_set_style_border_color(slices_container, lv_palettes_main(LV_PALETTE_BLUE), LV_PART_MAIN);
    lv_obj_set_style_bg_opa(slices_container, LV_OPA_TRANSP, LV_PART_MAIN);
    lv_obj_remove_flag(slices_container, LV_OBJ_FLAG_SCROLLABLE);

    /* Create slices */
    angle_accum = 0.0f;
    create_slice(slices_container, 12, lv_palettes_main(LV_PALETTE_RED));
    create_slice(slices_container, 18, lv_palettes_main(LV_PALETTE_BLUE));
    create_slice(slices_container, 26, lv_palettes_main(LV_PALETTE_GREEN));
    create_slice(slices_container, 24, lv_palettes_main(LV_PALETTE_ORANGE));
    create_slice(slices_container, 20, lv_palettes_main(LV_PALETTE_BLUE_GREY));
}
#endif

```

### 3.7.4 Barra

#### Barra semplice

```

#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

void lv_example_bar_1(void)
{
    lv_obj_t * bar1 = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar1, 200, 20);
    lv_obj_center(bar1);
    lv_bar_set_value(bar1, 70, LV_ANIM_OFF);
}

#endif

```

#### Stile di una barra

```

#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/** 
 * Example of styling the bar
 */
void lv_example_bar_2(void)
{
    static lv_style_t style_bg;
    static lv_style_t style_indic;

    lv_style_init(&style_bg);
    lv_style_set_border_color(&style_bg, lv_palettes_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style_bg, 2);
    lv_style_set_pad_all(&style_bg, 6); /*To make the indicator smaller*/
    lv_style_set_radius(&style_bg, 6);
    lv_style_set_anim_duration(&style_bg, 1000);

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palettes_main(LV_PALETTE_BLUE));
    lv_style_set_radius(&style_indic, 3);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * bar = lv_bar_create(lv_screen_active());
lv_obj_remove_style_all(bar); /*To have a clean start*/
lv_obj_add_style(bar, &style_bg, 0);
lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

lv_obj_set_size(bar, 200, 20);
lv_obj_center(bar);
lv_bar_set_value(bar, 100, LV_ANIM_ON);
}

#endif

```

## Indicatore di temperatura

```

#include "../../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

static void set_temp(void * bar, int32_t temp)
{
    lv_bar_set_value((lv_obj_t *)bar, temp, LV_ANIM_ON);
}

/**
 * A temperature meter example
 */
void lv_example_bar_3(void)
{
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_VER);

    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);
    lv_obj_set_size(bar, 20, 200);
    lv_obj_center(bar);
    lv_bar_set_range(bar, -20, 40);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_exec_cb(&a, set_temp);
    lv_anim_set_duration(&a, 3000);
    lv_anim_set_reverse_duration(&a, 3000);
    lv_anim_set_var(&a, bar);
    lv_anim_set_values(&a, -20, 40);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}

#endif

```

## Motivo a strisce e valore dell'intervallo

```

#include "../../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/**
 * Bar with stripe pattern and ranged value
 */
void lv_example_bar_4(void)
{
    LV_IMAGE_DECLARE(img_skew_strip);
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_image_src(&style_indic, &img_skew_strip);
    lv_style_set_bg_image_tiled(&style_indic, true);
    lv_style_set_bg_image_opa(&style_indic, LV_OPA_30);

    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

    lv_obj_set_size(bar, 260, 20);
    lv_obj_center(bar);
    lv_bar_set_mode(bar, LV_BAR_MODE_RANGE);
    lv_bar_set_value(bar, 90, LV_ANIM_OFF);
    lv_bar_set_start_value(bar, 20, LV_ANIM_OFF);
}

#endif

```

## Barra con direzione di base LTR e RTL

```

#include "../../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/**
 * Bar with LTR and RTL base direction
 */
void lv_example_bar_5(void)
{
    lv_obj_t * label;

    lv_obj_t * bar_ltr = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar_ltr, 200, 20);
    lv_bar_set_value(bar_ltr, 70, LV_ANIM_OFF);
    lv_obj_align(bar_ltr, LV_ALIGN_CENTER, 0, -30);

```

(continues on next page)

(continua dalla pagina precedente)

```

label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "Left to Right base direction");
lv_obj_align_to(label, bar_ltr, LV_ALIGN_OUT_TOP_MID, 0, -5);

lv_obj_t * bar_rtl = lv_bar_create(lv_screen_active());
lv_obj_set_style_base_dir(bar_rtl, LV_BASE_DIR_RTL, 0);
lv_obj_set_size(bar_rtl, 200, 20);
lv_bar_set_value(bar_rtl, 70, LV_ANIM_OFF);
lv_obj_align(bar_rtl, LV_ALIGN_CENTER, 0, 30);

label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "Right to Left base direction");
lv_obj_align_to(label, bar_rtl, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif

```

## Cassetto personalizzato per visualizzare il valore corrente

```

#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

#define MAX_VALUE 100
#define MIN_VALUE 0

static void set_value(void * bar, int32_t v)
{
    lv_bar_set_value((lv_obj_t *)bar, v, LV_ANIM_OFF);
}

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.font = LV_FONT_DEFAULT;

    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d", (int)lv_bar_get_value(obj));

    lv_text_attributes_t attributes = {0};
    attributes.letter_space = label_dsc.letter_space;
    attributes.line_space = label_dsc.line_space;
    attributes.max_width = LV_COORD_MAX;
    attributes.text_flags = label_dsc.flag;

    lv_point_t txt_size;
    lv_text_get_size(&txt_size, buf, label_dsc.font, &attributes);

    lv_area_t txt_area;
    txt_area.x1 = 0;
    txt_area.x2 = txt_size.x - 1;
    txt_area.y1 = 0;
    txt_area.y2 = txt_size.y - 1;

    lv_area_t indic_area;
    lv_obj_get_coords(obj, &indic_area);
    lv_area_set_width(&indic_area, lv_area_get_width(&indic_area) * lv_bar_get_value(obj) / MAX_VALUE);

    /*If the indicator is long enough put the text inside on the right*/
    if(lv_area_get_width(&indic_area) > txt_size.x + 20) {
        lv_area_align(&indic_area, &txt_area, LV_ALIGN_RIGHT_MID, -10, 0);
        label_dsc.color = lv_color_white();
    }
    /*If the indicator is still short put the text out of it on the right*/
    else {
        lv_area_align(&indic_area, &txt_area, LV_ALIGN_OUT_RIGHT_MID, 10, 0);
        label_dsc.color = lv_color_black();
    }
    label_dsc.text = buf;
    label_dsc.text_local = true;
    lv_layer_t * layer = lv_event_get_layer(e);
    lv_draw_label(layer, &label_dsc, &txt_area);
}

/**
 * Custom drawer on the bar to display the current value
 */
void lv_example_bar_6(void)
{
    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_bar_set_range(bar, MIN_VALUE, MAX_VALUE);
    lv_obj_set_size(bar, 200, 20);
    lv_obj_center(bar);
    lv_obj_add_event_cb(bar, event_cb, LV_EVENT_DRAW_MAIN_END, NULL);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, bar);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_exec_cb(&a, set_value);
    lv_anim_set_duration(&a, 4000);
    lv_anim_set_reverse_duration(&a, 4000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}
#endif

```

## Barra con direzione opposta

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/***
 * Bar with opposite direction
 */
void lv_example_bar_7(void)
{
    lv_obj_t * bar_tob = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar_tob, 20, 180);
    lv_bar_set_range(bar_tob, 100, 0);
    lv_bar_set_value(bar_tob, 70, LV_ANIM_OFF);
    lv_obj_align(bar_tob, LV_ALIGN_CENTER, 0, 10);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "From top to bottom");
    lv_obj_align_to(label, bar_tob, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif
```

## 3.7.5 Pulsante

### Pulsanti semplici

```
#include "../../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);

    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked");
    } else if(code == LV_EVENT_VALUE_CHANGED) {
        LV_LOG_USER("Toggled");
    }
}

void lv_example_button_1(void)
{
    lv_obj_t * label;

    lv_obj_t * btn1 = lv_button_create(lv_screen_active());
    lv_obj_add_event_cb(btn1, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -40);
    lv_obj_remove_flag(btn1, LV_OBJ_FLAG_PRESS_LOCK);

    label = lv_label_create(btn1);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);

    lv_obj_t * btn2 = lv_button_create(lv_screen_active());
    lv_obj_add_event_cb(btn2, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn2, LV_ALIGN_CENTER, 0, 40);
    lv_obj_add_flag(btn2, LV_OBJ_FLAG_CHECKABLE);
    lv_obj_set_height(btn2, LV_SIZE_CONTENT);

    label = lv_label_create(btn2);
    lv_label_set_text(label, "Toggle");
    lv_obj_center(label);
}

#endif
```

### Pulsanti con stili

```
#include "../../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

/***
 * Style a button from scratch
 */
void lv_example_button_2(void)
{
    /*Init the style for the default state*/
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 3);

    lv_style_set_bg_opa(&style, LV_OPA_100);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 2));
    lv_style_set_bg_grad_dir(&style, LV_GRAD_DIR_VER);

    lv_style_set_border_opa(&style, LV_OPA_40);
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_GREY));

    lv_style_set_shadow_width(&style, 8);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_shadow_offset_y(&style, 8);

    lv_style_set_outline_opa(&style, LV_OPA_COVER);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    lv_style_set_text_color(&style, lv_color_white());
    lv_style_set_pad_all(&style, 10);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Init the pressed style*/
static lv_style_t style_pr;
lv_style_init(&style_pr);

/*Add a large outline when pressed*/
lv_style_set_outline_width(&style_pr, 30);
lv_style_set_outline_opa(&style_pr, LV_OPA_TRANSP);

lv_style_set_translate_y(&style_pr, 5);
lv_style_set_shadow_offset_y(&style_pr, 3);
lv_style_set_bg_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 2));
lv_style_set_bg_grad_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 4));

/*Add a transition to the outline*/
static lv_style_transition_dsc_t trans;
static lv_style_prop_t props[] = {LV_STYLE_OUTLINE_WIDTH, LV_STYLE_OUTLINE_OPA, 0};
lv_style_transition_dsc_init(&trans, props, lv_anim_path_linear, 300, 0, NULL);

lv_style_set_transition(&style_pr, &trans);

lv_obj_t * btn1 = lv_button_create(lv_screen_active()); /*Remove the style coming from the theme*/
lv_obj_remove_style_all(btn1);
lv_obj_add_style(btn1, &style, 0);
lv_obj_add_style(btn1, &style_pr, LV_STATE_PRESSED);
lv_obj_set_size(btn1, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_center(btn1);

lv_obj_t * label = lv_label_create(btn1);
lv_label_set_text(label, "Button");
lv_obj_center(label);
}
#endif

```

## Pulsante gommoso

```

#include "../../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON

/**
 * Create a style transition on a button to act like a gum when clicked
 */
void lv_example_button_3(void)
{
    /*Properties to transition*/
    static lv_style_prop_t props[] = {
        LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_TRANSFORM_HEIGHT, LV_STYLE_TEXT_LETTER_SPACE, 0
    };

    /*Transition descriptor when going back to the default state.
     *Add some delay to be sure the press transition is visible even if the press was very short*/
    static lv_style_transition_dsc_t transition_dsc_def;
    lv_style_transition_dsc_init(&transition_dsc_def, props, lv_anim_path_overshoot, 250, 100, NULL);

    /*Transition descriptor when going to pressed state.
     *No delay, go to presses state immediately*/
    static lv_style_transition_dsc_t transition_dsc_pr;
    lv_style_transition_dsc_init(&transition_dsc_pr, props, lv_anim_path_ease_in_out, 250, 0, NULL);

    /*Add only the new transition to he default state*/
    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_transition(&style_def, &transition_dsc_def);

    /*Add the transition and some transformation to the presses state.*/
    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_transform_width(&style_pr, 10);
    lv_style_set_transform_height(&style_pr, -10);
    lv_style_set_text_letter_space(&style_pr, 10);
    lv_style_set_transition(&style_pr, &transition_dsc_pr);

    lv_obj_t * btn1 = lv_button_create(lv_screen_active());
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -80);
    lv_obj_add_style(btn1, &style_pr, LV_STATE_PRESSED);
    lv_obj_add_style(btn1, &style_def, 0);

    lv_obj_t * label = lv_label_create(btn1);
    lv_label_set_text(label, "Gum");
}
#endif

```

## 3.7.6 Matrice di Pulsanti

### Matrice di pulsanti semplice

```

#include "../../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        uint32_t id = lv_buttonmatrix_get_selected_button(obj);
        const char * txt = lv_buttonmatrix_get_button_text(obj, id);
        LV_UNUSED(txt);
        LV_LOG_USER("%s was pressed\n", txt);
    }
}

static const char * btmn_map[] = {"1", "2", "3", "4", "5", "\n",
                                 "6", "7", "8", "9", "0", "\n"};

```

(continues on next page)

(continua dalla pagina precedente)

```

        "Action1", "Action2", ""
    };

void lv_example_buttonmatrix_1(void)
{
    lv_obj_t * btm1 = lv_buttonmatrix_create(lv_screen_active());
    lv_buttonmatrix_set_map(btm1, btm_map);
    lv_buttonmatrix_set_button_width(btm1, 10, 2);           /*Make "Action1" twice as wide as "Action2"*/
    lv_buttonmatrix_set_button_ctrl(btm1, 10, LV_BUTTONMATRIX_CTRL_CHECKABLE);
    lv_buttonmatrix_set_button_ctrl(btm1, 11, LV_BUTTONMATRIX_CTRL_CHECKED);
    lv_obj_align(btm1, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(btm1, event_handler, LV_EVENT_ALL, NULL);
}

#endif

```

## Pulsanti personalizzati

```

#include "../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_event_get_draw_dsc(draw_task);
    /*When the button matrix draws the buttons...*/
    if(base_dsc->part == LV_PART_ITEMS) {
        bool pressed = false;
        if(lv_buttonmatrix_get_selected_button(obj) == base_dsc->id1 && lv_obj_has_state(obj, LV_STATE_PRESSED)) {
            pressed = true;
        }

        /*Change the draw descriptor of the 2nd button*/
        if(base_dsc->id1 == 1) {
            lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
            if(fill_draw_dsc) {
                fill_draw_dsc->radius = 0;
                if(pressed) fill_draw_dsc->color = lv_palette_darken(LV_PALETTE_BLUE, 3);
                else fill_draw_dsc->color = lv_palette_main(LV_PALETTE_BLUE);
            }
            lv_draw_box_shadow_dsc_t * box_shadow_draw_dsc = lv_draw_task_get_box_shadow_dsc(draw_task);
            if(box_shadow_draw_dsc) {
                box_shadow_draw_dsc->width = 6;
                box_shadow_draw_dsc->ofs_x = 3;
                box_shadow_draw_dsc->ofs_y = 3;
            }
            lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
            if(label_draw_dsc) {
                label_draw_dsc->color = lv_color_white();
            }
        }

        /*Change the draw descriptor of the 3rd button*/
        else if(base_dsc->id1 == 2) {
            lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
            if(fill_draw_dsc) {
                fill_draw_dsc->radius = LV_RADIUS_CIRCLE;
                if(pressed) fill_draw_dsc->color = lv_palette_darken(LV_PALETTE_RED, 3);
                else fill_draw_dsc->color = lv_palette_main(LV_PALETTE_RED);
            }
            lv_draw_box_shadow_dsc_t * box_shadow_draw_dsc = lv_draw_task_get_box_shadow_dsc(draw_task);
            if(box_shadow_draw_dsc) {
                box_shadow_draw_dsc->radius = LV_RADIUS_CIRCLE;
            }
        }
        else if(base_dsc->id1 == 3) {
            lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
            if(label_draw_dsc) {
                label_draw_dsc->opa = 0;
            }
            if(lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_FILL) {
                LV_IMAGE_DECLARE(img_star);
                lv_image_header_t header;
                lv_result_t res = lv_image_decoder_get_info(&img_star, &header);
                if(res != LV_RESULT_OK) return;

                lv_area_t a;
                a.x1 = 0;
                a.x2 = header.w - 1;
                a.y1 = 0;
                a.y2 = header.h - 1;
                lv_area_t draw_task_area;
                lv_draw_task_get_area(draw_task, &draw_task_area);
                lv_area_align(&draw_task_area, &a, LV_ALIGN_CENTER, 0, 0);

                lv_draw_image_dsc_t img_draw_dsc;
                lv_draw_image_dsc_init(&img_draw_dsc);
                img_draw_dsc.src = &img_star;
                img_draw_dsc.recolor = lv_color_black();
                if(pressed) img_draw_dsc.recolor_opa = LV_OPA_30;

                lv_draw_image(base_dsc->layer, &img_draw_dsc, &a);
            }
        }
    }
}

/**
 * Add custom drawer to the button matrix to customize buttons one by one
 */
void lv_example_buttonmatrix_2(void)
{
    lv_obj_t * btm = lv_buttonmatrix_create(lv_screen_active());
    lv_obj_add_event_cb(btm, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(btm, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_obj_center(btnm);
}

#endif

```

## Paginazione

```

#include "../../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    uint32_t id = lv_buttonmatrix_get_selected_button(obj);
    bool prev = id == 0;
    bool next = id == 6;
    if(prev || next) {
        /*Find the checked button*/
        uint32_t i;
        for(i = 1; i < 7; i++) {
            if(lv_buttonmatrix_has_button_ctrl(obj, i, LV_BUTTONMATRIX_CTRL_CHECKED)) break;
        }
        if(prev && i > 1) i--;
        else if(next && i < 5) i++;
        lv_buttonmatrix_set_button_ctrl(obj, i, LV_BUTTONMATRIX_CTRL_CHECKED);
    }
}

/**
 * Make a button group (pagination)
 */
void lv_example_buttonmatrix_3(void)
{
    static lv_style_t style_bg;
    lv_style_init(&style_bg);
    lv_style_set_pad_all(&style_bg, 0);
    lv_style_set_pad_gap(&style_bg, 0);
    lv_style_set_clip_corner(&style_bg, true);
    lv_style_set_radius(&style_bg, LV_RADIUS_CIRCLE);
    lv_style_set_border_width(&style_bg, 0);

    static lv_style_t style_btn;
    lv_style_init(&style_btn);
    lv_style_set_radius(&style_btn, 0);
    lv_style_set_border_width(&style_btn, 1);
    lv_style_set_border_opa(&style_btn, LV_OPA_50);
    lv_style_set_border_color(&style_btn, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_border_side(&style_btn, LV_BORDER_SIDE_INTERNAL);
    lv_style_set_radius(&style_btn, 0);

    static const char * map[] = {LV_SYMBOL_LEFT, "1", "2", "3", "4", "5", LV_SYMBOL_RIGHT, ""};

    lv_obj_t * btnm = lv_buttonmatrix_create(lv_screen_active());
    lv_buttonmatrix_set_map(btnm, map);
    lv_obj_add_style(btnm, &style_bg, LV_PART_ITEMS);
    lv_obj_add_event_cb(btnm, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_size(btnm, 225, 35);

    /*Allow selecting on one number at time*/
    lv_buttonmatrix_set_button_ctrl_all(btnm, LV_BUTTONMATRIX_CTRL_CHECKABLE);
    lv_buttonmatrix_clear_button_ctrl(btnm, 0, LV_BUTTONMATRIX_CTRL_CHECKABLE);
    lv_buttonmatrix_clear_button_ctrl(btnm, 6, LV_BUTTONMATRIX_CTRL_CHECKABLE);

    lv_buttonmatrix_set_one_checked(btnm, true);
    lv_buttonmatrix_set_button_ctrl(btnm, 1, LV_BUTTONMATRIX_CTRL_CHECKED);

    lv_obj_center(btnm);
}
#endif

```

## 3.7.7 Calendario

### Calendario con intestazione

```

#include "../../../lv_examples.h"
#if LV_USE_CALENDAR && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = (lv_obj_t *)lv_event_get_current_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_calendar_date_t date;
        if(lv_calendar_get_pressed_date(obj, &date)) {
            LV_LOG_USER("Clicked date: %02d.%02d.%d", date.day, date.month, date.year);
        }
    }
}

void lv_example_calendar_1(void)
{
    lv_obj_t * calendar = lv_calendar_create(lv_screen_active());
    lv_obj_set_size(calendar, 185, 230);
    lv_obj_align(calendar, LV_ALIGN_CENTER, 0, 27);
    lv_obj_add_event_cb(calendar, event_handler, LV_EVENT_ALL, NULL);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_calendar_set_today_date(calendar, 2021, 02, 23);
lv_calendar_set_month_shown(calendar, 2021, 02);

/*Highlight a few days/
static lv_calendar_date_t highlighted_days[3];      /*Only its pointer will be saved so should be static*/
highlighted_days[0].year = 2021;
highlighted_days[0].month = 02;
highlighted_days[0].day = 6;

highlighted_days[1].year = 2021;
highlighted_days[1].month = 02;
highlighted_days[1].day = 11;

highlighted_days[2].year = 2022;
highlighted_days[2].month = 02;
highlighted_days[2].day = 22;

lv_calendar_set_highlighted_dates(calendar, highlighted_days, 3);

#if LV_USE_CALENDAR_HEADER_DROPDOWN
lv_calendar_add_header_dropdown(calendar);
#elif LV_USE_CALENDAR_HEADER_ARROW
lv_calendar_add_header_arrow(calendar);
#endif
}

#endif
}

```

## Calendario cinese

```

#include "../../lv_examples.h"
#if LV_USE_CALENDAR && LV_USE_CALENDAR_CHINESE && LV_BUILD_EXAMPLES

void lv_example_calendar_2(void)
{
    lv_obj_t * calendar = lv_calendar_create(lv_screen_active());
    lv_obj_set_size(calendar, 300, 300);
    lv_obj_align(calendar, LV_ALIGN_TOP_MID, 0, 0);

    lv_calendar_set_today_date(calendar, 2024, 03, 22);
    lv_calendar_set_month_shown(calendar, 2024, 03);

#if LV_USE_CALENDAR_HEADER_DROPDOWN
lv_calendar_add_header_dropdown(calendar);
#elif LV_USE_CALENDAR_HEADER_ARROW
lv_calendar_add_header_arrow(calendar);
#endif

    lv_calendar_set_chinese_mode(calendar, true);
    lv_obj_set_style_text_font(calendar, &lv_font_source_han_sans_sc_14_cjk, LV_PART_MAIN);
}

#else

void lv_example_calendar_2(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "chinese calendar is not enabled");
    lv_obj_center(label);
}

#endif
}

```

## 3.7.8 Canvas

### Disegno sulla Canvas e rotazione

```

#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 200
#define CANVAS_HEIGHT 150

void lv_example_canvas_1(void)
{
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.radius = 10;
    rect_dsc.bg_opa = LV_OPA_COVER;
    rect_dsc.bg_grad.dir = LV_GRAD_DIR_VER;
    rect_dsc.bg_grad.stops[0].color = lv_palette_main(LV_PALETTE_RED);
    rect_dsc.bg_grad.stops[0].opa = LV_OPA_100;
    rect_dsc.bg_grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
    rect_dsc.bg_grad.stops[1].opa = LV_OPA_50;
    rect_dsc.border_width = 2;
    rect_dsc.border_opa = LV_OPA_90;
    rect_dsc.border_color = lv_color_white();
    rect_dsc.shadow_width = 5;
    rect_dsc.shadow_offset_x = 5;
    rect_dsc.shadow_offset_y = 5;

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_palette_main(LV_PALETTE_ORANGE);
    label_dsc.text = "Some text on text canvas";
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf_16bpp, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_RGB565);
    LV_DRAW_BUF_INIT_STATIC(draw_buf_16bpp);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf_16bpp);
    lv_obj_center(canvas);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);

lv_layer_t layer;
lv_canvas_init_layer(canvas, &layer);

lv_area_t coords_rect = {30, 20, 100, 70};
lv_draw_rect(&layer, &rect_dsc, &coords_rect);

lv_area_t coords_text = {40, 80, 100, 120};
lv_draw_label(&layer, &label_dsc, &coords_text);

lv_canvas_finish_layer(canvas, &layer);

/*Test the rotation. It requires another buffer where the original image is stored.
 *So use previous canvas as image and rotate it to the new canvas*/
LV_DRAW_BUF_DEFINE_STATIC(draw_buf_32bpp, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
LV_DRAW_BUF_INIT_STATIC(draw_buf_32bpp);

/*Create a canvas and initialize its palette*/
canvas = lv_canvas_create(lv_screen_active());
lv_canvas_set_draw_buf(canvas, &draw_buf_32bpp);
lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
lv_obj_center(canvas);

lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 1), LV_OPA_COVER);

lv_canvas_init_layer(canvas, &layer);
lv_image_dsc_t img;
lv_draw_buf_to_image(&draw_buf_16bpp, &img);
lv_draw_image_dsc_t img_dsc;
lv_draw_image_dsc_init(&img_dsc);
img_dsc.rotation = 120;
img_dsc.src = &img;
img_dsc.pivot.x = CANVAS_WIDTH / 2;
img_dsc.pivot.y = CANVAS_HEIGHT / 2;

lv_area_t coords_img = {0, 0, CANVAS_WIDTH - 1, CANVAS_HEIGHT - 1};
lv_draw_image(&layer, &img_dsc, &coords_img);

lv_canvas_finish_layer(canvas, &layer);
}

#endif

```

## Canvas trasparente con chroma key

```

#include "../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 80
#define CANVAS_HEIGHT 40

/**
 * Create a transparent canvas with transparency
 */
void lv_example_canvas_2(void)
{
    lv_obj_set_style_bg_color(lv_screen_active(), lv_palette_lighten(LV_PALETTE_RED, 5), 0);

    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_obj_center(canvas);

    /*Red background (There is no dedicated alpha channel in indexed images so LV_OPA_COVER is ignored)*/
    lv_canvas_fill_bg(canvas, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_COVER);

    /*Create hole on the canvas*/
    int32_t x;
    int32_t y;
    for(y = 10; y < 20; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_50);
        }
    }

    for(y = 20; y < 30; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_20);
        }
    }

    for(y = 30; y < 40; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_0);
        }
    }
}
#endif

```

## Disegnare un rettangolo sul canvas

```

#include "../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/**
 * Draw a rectangle to the canvas
 */

```

(continues on next page)

(continua dalla pagina precedente)

```
/*
void lv_example_canvas_3(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);

    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_rect_dsc_t dsc;
    lv_draw_rect_dsc_init(&dsc);
    dsc.bg_color = lv_palette_main(LV_PALETTE_RED);
    dsc.border_color = lv_palette_main(LV_PALETTE_BLUE);
    dsc.border_width = 3;
    dsc.outline_color = lv_palette_main(LV_PALETTE_GREEN);
    dsc.outline_width = 2;
    dsc.outline_pad = 2;
    dsc.outline_opa = LV_OPA_50;
    dsc.radius = 5;
    dsc.border_width = 3;

    lv_area_t coords = {10, 10, 40, 30};

    lv_draw_rect(&layer, &dsc, &coords);
    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

## Disegnare un'etichetta sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_FONT_MONTserrat_18 && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a text to the canvas
 */
void lv_example_canvas_4(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_label_dsc_t dsc;
    lv_draw_label_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.font = &lv_font_montserrat_18;
    dsc.decor = LV_TEXT_DECOR_UNDERLINE;
    dsc.text = "Hello";

    lv_area_t coords = {10, 10, 30, 60};

    lv_draw_label(&layer, &dsc, &coords);
    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

## Disegnare un arco sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw an arc to the canvas
 */
void lv_example_canvas_5(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_draw_arc_dsc_t dsc;
lv_draw_arc_dsc_init(&dsc);
dsc.color = lv_palette_main(LV_PALETTE_RED);
dsc.width = 5;
dsc.center.x = 25;
dsc.center.y = 25;
dsc.width = 10;
dsc.radius = 15;
dsc.start_angle = 0;
dsc.end_angle = 220;

lv_draw_arc(&layer, &dsc);

lv_canvas_finish_layer(canvas, &layer);

}
#endif

```

## Disegnare un'immagine sul canvas

```

#include "../../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw an image to the canvas
 */
void lv_example_canvas_6(void)
{
    /*Create a buffer for the canvas*/
    static uint8_t cbuf[LV_CANVAS_BUF_SIZE(CANVAS_WIDTH, CANVAS_HEIGHT, 32, LV_DRAW_BUF_STRIDE_ALIGN)];

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    LV_IMAGE_DECLARE(img_star);
    lv_draw_image_dsc_t dsc;
    lv_draw_image_dsc_init(&dsc);
    dsc.src = &img_star;

    lv_area_t coords = {10, 10, 10 + img_star.header.w - 1, 10 + img_star.header.h - 1};

    lv_draw_image(&layer, &dsc, &coords);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif

```

## Disegnare una linea sul canvas

```

#include "../../../lv_examples.h"
#if LV_USE_CANVAS&& LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a line to the canvas
 */
void lv_example_canvas_7(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_line_dsc_t dsc;
    lv_draw_line_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.width = 4;
    dsc.round_end = 1;
    dsc.round_start = 1;
    dsc.p1.x = 15;
    dsc.p1.y = 15;
    dsc.p2.x = 35;
    dsc.p2.y = 10;
    lv_draw_line(&layer, &dsc);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif

```

## Disegnare un'immagine vettoriale sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#if LV_USE_VECTOR_GRAPHIC

#define CANVAS_WIDTH 150
#define CANVAS_HEIGHT 150

/***
 * Draw a path to the canvas
 */
void lv_example_canvas_8(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_vector_dsc_t * dsc = lv_vector_dsc_create(&layer);
    lv_vector_path_t * path = lv_vector_path_create(LV_VECTOR_PATH_QUALITY_MEDIUM);

    lv_fpoint_t pts[] = {{10, 10}, {130, 130}, {10, 130}};
    lv_vector_path_move_to(path, &pts[0]);
    lv_vector_path_line_to(path, &pts[1]);
    lv_vector_path_line_to(path, &pts[2]);
    lv_vector_path_close(path);

    lv_vector_dsc_set_fill_color(dsc, lv_color_make(0x00, 0x80, 0xff));
    lv_vector_dsc_add_path(dsc, path);

    lv_draw_vector(dsc);
    lv_vector_path_delete(path);
    lv_vector_dsc_delete(dsc);

    lv_canvas_finish_layer(canvas, &layer);
}
#else

void lv_example_canvas_8(void)
{
    /*fallback for online examples*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Vector graphics is not enabled");
    lv_obj_center(label);
}

#endif /*LV_USE_VECTOR_GRAPHIC*/
#endif

```

## Disegnare un triangolo sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 150
#define CANVAS_HEIGHT 150

/***
 * Draw a triangle to the canvas
 */
void lv_example_canvas_9(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_triangle_dsc_t tri_dsc;
    lv_draw_triangle_dsc_init(&tri_dsc);
    tri_dsc.p[0].x = 10;
    tri_dsc.p[0].y = 10;
    tri_dsc.p[1].x = 100;
    tri_dsc.p[1].y = 30;
    tri_dsc.p[2].x = 50;
    tri_dsc.p[2].y = 100;

    tri_dsc.grad.stops_count = 2;
    tri_dsc.grad.dir = LV_GRAD_DIR_VER;
    tri_dsc.grad.stops[0].color = lv_color_hex(0xffff00);
    tri_dsc.grad.stops[0].frac = 64; /*Start at 25%*/
    tri_dsc.grad.stops[0].opa = LV_OPA_COVER;
    tri_dsc.grad.stops[1].color = lv_color_hex(0x00ffff);
    tri_dsc.grad.stops[1].opa = LV_OPA_TRANSP;
    tri_dsc.grad.stops[1].frac = 3 * 64; /*End at 75%*/

    tri_dsc.opa = 128; /*Set the overall opacity to 50%*/

    lv_draw_triangle(&layer, &tri_dsc);
}
```

(continues on next page)

(continua dalla pagina precedente)

```
lv_canvas_finish_layer(canvas, &layer);
}

#endif
```

## Disegnare Effetti di Lettere Fantasiose

```
#include "../../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 300
#define CANVAS_HEIGHT 200

static void timer_cb(lv_timer_t * timer)
{
    static int16_t counter = 0;
    const char * string = "lol- I'm wavvvvvvvving->>>";
    const int16_t string_len = lv_strlen(string);

    lv_obj_t * canvas = (lv_obj_t *)lv_timer_get_user_data(timer);
    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_COVER);

    lv_draw_letter_dsc_t letter_dsc;
    lv_draw_letter_dsc_init(&letter_dsc);
    letter_dsc.color = lv_color_hex(0xffff0000);
    letter_dsc.font = lv_font_get_default();

    {
#define CURVE2_X(t) (t * 2 + 10)
#define CURVE2_Y(t) (lv_trigo_sin((t) * 5) * 40 / 32767 + CANVAS_HEIGHT / 2)

        int32_t pre_x = CURVE2_X(-1);
        int32_t pre_y = CURVE2_Y(-1);
        for(int16_t i = 0; i < string_len; i++) {
            const int16_t angle = (int16_t)(i * 5);
            const int32_t x = CURVE2_X(angle);
            const int32_t y = CURVE2_Y(angle + counter / 2);
            const lv_point_t point = { .x = x, .y = y };

            letter_dsc.unicode = (uint32_t)string[i % string_len];
            letter_dsc.rotation = lv_atan2(y - pre_y, x - pre_x) * 10;
            letter_dsc.color = lv_color_hsv_to_rgb(i * 10, 100, 100);
            lv_draw_letter(&layer, &letter_dsc, &point);

            pre_x = x;
            pre_y = y;
        }
    }

    lv_canvas_finish_layer(canvas, &layer);
    counter++;
}

void lv_example_canvas_10(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_obj_set_size(canvas, CANVAS_WIDTH, CANVAS_HEIGHT);

    lv_obj_center(canvas);

    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_timer_create(timer_cb, 16, canvas);
}
#endif
```

## Disegnare Effetti di Lettere Fantasiose 2

```
#include "../../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 300
#define CANVAS_HEIGHT 200

static void timer_cb(lv_timer_t * timer)
{
    static int32_t counter = 0;
    const char * string = "windstooooooooooooo->>>";
    const int16_t string_len = lv_strlen(string);

    lv_obj_t * canvas = (lv_obj_t *) lv_timer_get_user_data(timer);
    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_COVER);

    lv_draw_letter_dsc_t letter_dsc;
    lv_draw_letter_dsc_init(&letter_dsc);
    letter_dsc.color = lv_color_hex(0xffff0000);
    letter_dsc.font = lv_font_get_default();

    {
#define CURVE2_X(t) ((t) * 2 + lv_trigo_cos((t) * 5) * 40 / 32767 - 10)
#define CURVE2_Y(t, T) ((t) * lv_trigo_sin(((t) + (T)) * 5) * 40 / 32767 / 80 + CANVAS_HEIGHT / 2)
```

(continues on next page)

(continua dalla pagina precedente)

```

int32_t pre_x = CURVE2_X(-1);
int32_t pre_y = CURVE2_Y(-1, 0);
for(int16_t i = 0; i < string_len; i++) {
    const int32_t angle = i * 5;
    const int32_t x = CURVE2_X(angle);
    const int32_t y = CURVE2_Y(angle + 30, counter / 2);

    letter_dsc.unicode = (uint32_t)string[i % string_len];
    letter_dsc.rotation = lv_atan2(y - pre_y, x - pre_x) * 10;
    letter_dsc.color = lv_color_hsv_to_rgb(i * 10, 100, 100);
    lv_point_t p = (lv_point_t) {
        .x = x, .y = y
    };
    lv_draw_letter(&layer, &letter_dsc, &p);

    pre_x = x;
    pre_y = y;
}
}

lv_canvas_finish_layer(canvas, &layer);
counter++;
}

void lv_example_canvas_11(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_obj_set_size(canvas, CANVAS_WIDTH, CANVAS_HEIGHT);

    lv_obj_center(canvas);

    lv_canvas_set_draw_buf(canvas, &draw_buf);

    lv_timer_create(timer_cb, 16, canvas);
}
#endif

```

## 3.7.9 Grafico

### Grafico a linee

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

/***
 * A very basic line chart
 */
void lv_example_chart_1(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_center(chart);
    lv_chart_set_type(chart, LV_CHART_TYPE_LINE); /*Show lines and points too*/

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_GREEN), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_SECONDARY_Y);
    int32_t * ser2_y_points = lv_chart_get_series_y_array(chart, ser2);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        /*Set the next points on 'ser1'*/
        lv_chart_set_next_value(chart, ser1, (int32_t)lv_rand(10, 50));

        /*Directly set points on 'ser2'*/
        ser2_y_points[i] = (int32_t)lv_rand(50, 90);
    }
    lv_chart_refresh(chart); /*Required after direct set*/
}
#endif

```

### Segni di spunta e etichette degli assi con scorrimento su un grafico

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

/***
 * Use lv_scale to add ticks to a scrollable chart
 */
void lv_example_chart_2(void)
{
    /*Create a container*/
    lv_obj_t * main_cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(main_cont, 200, 150);
    lv_obj_center(main_cont);

    /*Create a transparent wrapper for the chart and the scale.
     *Set a large width, to make it scrollable on the main container*/
    lv_obj_t * wrapper = lv_obj_create(main_cont);
    lv_obj_remove_style_all(wrapper);
    lv_obj_set_size(wrapper, lv_pct(300), lv_pct(100));
    lv_obj_set_flex_flow(wrapper, LV_FLEX_FLOW_COLUMN);

```

(continues on next page)

(continua dalla pagina precedente)

```

/*Create a chart on the wrapper
*Set it's width to 100% to fill the large wrapper*/
lv_obj_t * chart = lv_chart_create(wrapper);
lv_obj_set_width(chart, lv_pct(100));
lv_obj_set_flex_grow(chart, 1);
lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 100);
lv_chart_set_axis_range(chart, LV_CHART_AXIS_SECONDARY_Y, 0, 400);
lv_chart_set_point_count(chart, 12);
lv_obj_set_style_radius(chart, 0, 0);

/*Create a scale also with 100% width*/
lv_obj_t * scale_bottom = lv_scale_create(wrapper);
lv_scale_set_mode(scale_bottom, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
lv_obj_set_size(scale_bottom, lv_pct(100), 25);
lv_scale_set_total_tick_count(scale_bottom, 12);
lv_scale_set_major_tick_every(scale_bottom, 1);
lv_obj_set_style_pad_hor(scale_bottom, lv_chart_get_first_point_center_offset(chart), 0);

static const char * month[] = {"Jan", "Feb", "March", "Apr", "May", "Jun", "July", "Aug", "Sept", "Oct", "Nov", "Dec", NULL};
lv_scale_set_text_src(scale_bottom, month);

/*Add two data series*/
lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_lighten(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);
lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_darker(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);

/*Set the next points on 'ser1'*/
uint32_t i;
for(i = 0; i < 12; i++) {
    lv_chart_set_next_value(chart, ser1, (int32_t)lv_rand(10, 60));
    lv_chart_set_next_value(chart, ser2, (int32_t)lv_rand(50, 90));
}
lv_chart_refresh(chart); /*Required after direct set*/
}

#endif

```

## Mostrare il valore dei punti premuti

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * chart = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_invalidate(chart);
    }
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        int32_t *s = (int32_t *)lv_event_get_param(e);
        *s = LV_MAX(*s, 20);
    }
    else if(code == LV_EVENT_DRAW_POST_END) {
        uint32_t id = lv_chart_get_pressed_point(chart);
        if(id == LV_CHART_POINT_NONE) return;

        LV_LOG_USER("Selected point %d", (int)id);

        lv_chart_series_t * ser = lv_chart_get_series_next(chart, NULL);
        int32_t value = 0;
        while(ser) {
            lv_point_t p;
            lv_chart_get_point_pos_by_id(chart, ser, id, &p);

            int32_t * y_array = lv_chart_get_series_y_array(chart, ser);
            if(y_array[id] != LV_CHART_POINT_NONE && y_array[id] >= 0) {

                /*Accumulate the values to show the rectangles at the top of each segment*/
                value += y_array[id];

                /*Draw a rectangle above the clicked point*/
                lv_layer_t * layer = lv_event_get_layer(e);
                lv_draw_rect_dsc_t draw_rect_dsc;
                lv_draw_rect_dsc_init(&draw_rect_dsc);
                draw_rect_dsc.bg_color = lv_color_black();
                draw_rect_dsc.bg_opa = LV_OPA_50;
                draw_rect_dsc.radius = 3;

                lv_area_t chart_obj_coords;
                lv_obj_get_coords(chart, &chart_obj_coords);
                lv_area_t rect_area;
                rect_area.x1 = chart_obj_coords.x1 + p.x - 20;
                rect_area.x2 = chart_obj_coords.x1 + p.x + 20;
                rect_area.y1 = chart_obj_coords.y1 + p.y - 10;
                rect_area.y2 = chart_obj_coords.y1 + p.y + 10;
                lv_draw_rect(layer, &draw_rect_dsc, &rect_area);

                /*Draw the value as label to the center of the rectangle*/
                char buf[16];
                lv_snprintf(buf, sizeof(buf), LV_SYMBOL_DUMMY"%d", value);

                lv_draw_label_dsc_t draw_label_dsc;
                lv_draw_label_dsc_init(&draw_label_dsc);
                draw_label_dsc.color = lv_color_white();
                draw_label_dsc.text = buf;
                draw_label_dsc.text_local = 1;
                draw_label_dsc.align = LV_TEXT_ALIGN_CENTER;
                lv_area_t label_area;
                lv_area_set_height(&label_area, lv_font_get_line_height(draw_label_dsc.font));
                lv_area_align(&rect_area, &label_area, LV_ALIGN_CENTER, 0, 0);
                lv_draw_label(layer, &draw_label_dsc, &label_area);
            }

            ser = lv_chart_get_series_next(chart, ser);
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    }
    else if(code == LV_EVENT_RELEASED) {
        lv_obj_invalidate(chart);
    }
}

/**
 * Show the value of the pressed points
 */
void lv_example_chart_3(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 280, 180);
    lv_obj_set_style_pad_column(chart, 4, 0);
    lv_obj_center(chart);
    lv_chart_set_type(chart, LV_CHART_TYPE_STACKED);

    lv_obj_add_event_cb(chart, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_GREEN), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser3 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_BLUE), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser1, lv_rand(15, 30));
        lv_chart_set_next_value(chart, ser2, lv_rand(15, 30));
        lv_chart_set_next_value(chart, ser3, lv_rand(15, 30));
    }
}
#endif

```

## Ricolorare le barre in base al loro valore

```

#include "../../../lv_examples.h"

#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_dsc(draw_task);

    if(base_dsc->part != LV_PART_ITEMS) {
        return;
    }

    lv_draw_fill_dsc_t * fill_dsc = lv_draw_task_get_fill_dsc(draw_task);
    if(fill_dsc) {
        lv_obj_t * chart = lv_event_get_target_obj(e);
        int32_t * y_array = lv_chart_get_series_y_array(chart, lv_chart_get_series_next(chart, NULL));
        int32_t v = y_array[base_dsc->i2];

        uint8_t ratio = (uint8_t)(v * 255 / 100);
        fill_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_GREEN), lv_palette_main(LV_PALETTE_RED), ratio);
    }
}

/**
 * Recolor the bars of a chart based on their value
 */
void lv_example_chart_4(void)
{
    /*Create a chart1*/
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
    lv_chart_set_point_count(chart, 24);
    lv_obj_set_style_pad_column(chart, 2, 0);
    lv_obj_set_size(chart, 260, 160);
    lv_obj_center(chart);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_color_hex(0xffff00), LV_CHART_AXIS_PRIMARY_Y);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(chart, LV_OB_FLAG_SEND_DRAW_TASK_EVENTS);

    uint32_t i;
    for(i = 0; i < 24; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
    }
}
#endif

```

## Grafico a linee sfumate con linee di divisione personalizzate

```

#include "../../../lv_examples.h"

#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES
#include "../../../lvgl_private.h"

static void hook_division_lines(lv_event_t * e);
static void add_faded_area(lv_event_t * e);
static void draw_event_cb(lv_event_t * e);

/**
 * Add a faded area effect to the line chart and make some division lines ticker
 */
void lv_example_chart_5(void)
{
    /*Create a chart*/

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * chart = lv_chart_create(lv_screen_active());
lv_chart_set_type(chart, LV_CHART_TYPE_LINE); /*Show lines and points too*/
lv_obj_set_size(chart, 200, 150);
lv_obj_set_style_pad_all(chart, 0, 0);
lv_obj_set_style_radius(chart, 0, 0);
lv_obj_center(chart);

lv_chart_set_div_line_count(chart, 5, 7);

lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
lv_obj_add_flag(chart, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);

uint32_t i;
for(i = 0; i < 10; i++) {
    lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 80));
}
}

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);

    if(base_dsc->part == LV_PART_ITEMS && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_LINE) {
        add_faded_area(e);
    }
    /*Hook the division lines too*/
    if(base_dsc->part == LV_PART_MAIN && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_LINE) {
        hook_division_lines(e);
    }
}

static void add_faded_area(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_area_t coords;
    lv_obj_get_coords(obj, &coords);

    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);

    const lv_chart_series_t * ser = lv_chart_get_series(next(obj, NULL));
    lv_color_t ser_color = lv_chart_get_series_color(obj, ser);

    /*Draw a triangle below the line with some opacity gradient/
    lv_draw_line_dsc_t * draw_line_dsc = (lv_draw_line_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_triangle_dsc_t tri_dsc;

    lv_draw_triangle_dsc_init(&tri_dsc);
    tri_dsc.p[0].x = draw_line_dsc->p1.x;
    tri_dsc.p[0].y = draw_line_dsc->p1.y;
    tri_dsc.p[1].x = draw_line_dsc->p2.x;
    tri_dsc.p[1].y = draw_line_dsc->p2.y;
    tri_dsc.p[2].x = draw_line_dsc->p1.y < draw_line_dsc->p2.y ? draw_line_dsc->p1.x : draw_line_dsc->p2.x;
    tri_dsc.p[2].y = LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y);
    tri_dsc.grad.dir = LV_GRAD_DIR_VER;

    int32_t full_h = lv_obj_get_height(obj);
    int32_t fract_uppter = (int32_t)(LV_MIN(draw_line_dsc->p1.y, draw_line_dsc->p2.y) - coords.y1) * 255 / full_h;
    int32_t fract_lower = (int32_t)(LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y) - coords.y1) * 255 / full_h;
    tri_dsc.grad.stops[0].color = ser_color;
    tri_dsc.grad.stops[0].opa = (lv_opa_t)(255 - fract_uppter);
    tri_dsc.grad.stops[0].frac = 0;
    tri_dsc.grad.stops[1].color = ser_color;
    tri_dsc.grad.stops[1].opa = (lv_opa_t)(255 - fract_lower);
    tri_dsc.grad.stops[1].frac = 255;

    lv_draw_triangle(base_dsc->layer, &tri_dsc);

    /*Draw rectangle below the triangle*/
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.bg_grad.dir = LV_GRAD_DIR_VER;
    rect_dsc.bg_grad.stops[0].color = ser_color;
    rect_dsc.bg_grad.stops[0].frac = 0;
    rect_dsc.bg_grad.stops[0].opa = (lv_opa_t)(255 - fract_lower);
    rect_dsc.bg_grad.stops[1].color = ser_color;
    rect_dsc.bg_grad.stops[1].frac = 255;
    rect_dsc.bg_grad.stops[1].opa = 0;

    lv_area_t rect_area;
    rect_area.x1 = (int32_t)draw_line_dsc->p1.x;
    rect_area.x2 = (int32_t)draw_line_dsc->p2.x - 1;
    rect_area.y1 = (int32_t)LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y) - 1;
    rect_area.y2 = (int32_t)coords.y2;
    lv_draw_rect(base_dsc->layer, &rect_dsc, &rect_area);
}

static void hook_division_lines(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_line_dsc_t * line_dsc = (lv_draw_line_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);

    /*Vertical line*/
    if(line_dsc->p1.x == line_dsc->p2.x) {
        line_dsc->color = lv_palette_lighten(LV_PALETTE_GREY, 1);
        if(base_dsc->id1 == 3) {
            line_dsc->width = 2;
            line_dsc->dash_gap = 0;
            line_dsc->dash_width = 0;
        }
        else {
            line_dsc->width = 1;
            line_dsc->dash_gap = 6;
            line_dsc->dash_width = 6;
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/*Horizontal line*/
else {
    if(base_dsc->id1 == 2) {
        line_dsc->width = 2;
        line_dsc->dash_gap = 0;
        line_dsc->dash_width = 0;
    }
    else {
        line_dsc->width = 2;
        line_dsc->dash_gap = 6;
        line_dsc->dash_width = 6;
    }

    if(base_dsc->id1 == 1 || base_dsc->id1 == 3) {
        line_dsc->color = lv_palette_main(LV_PALETTE_GREEN);
    }
    else {
        line_dsc->color = lv_palette_lighten(LV_PALETTE_GREY, 1);
    }
}
#endif

```

## Mostrare il cursore sul punto cliccato

```

#include "../../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static lv_obj_t * chart;
static lv_chart_series_t * ser;
static lv_chart_cursor_t * cursor;

static void value_changed_event_cb(lv_event_t * e)
{
    uint32_t last_id;
    lv_obj_t * obj = lv_event_get_target_obj(e);

    last_id = lv_chart_get_pressed_point(obj);
    if(last_id != LV_CHART_POINT_NONE) {
        lv_chart_set_cursor_point(obj, cursor, NULL, last_id);
    }
}

/**
 * Show cursor on the clicked point
 */
void lv_example_chart_6(void)
{
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, -10);

    // lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_Y, 10, 5, 6, 5, true, 40);
    // lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_X, 10, 5, 10, 1, true, 30);

    lv_obj_add_event_cb(chart, value_changed_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    cursor = lv_chart_add_cursor(chart, lv_palette_main(LV_PALETTE_BLUE), (lv_dir_t)(LV_DIR_LEFT | LV_DIR_BOTTOM));

    ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
    }

    // lv_chart_set_scale_x(chart, 500);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Click on a point");
    lv_obj_align_to(label, chart, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif

```

## Grafico a dispersione

```

#include "../../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        lv_obj_t * obj = lv_event_get_target_obj(e);
        lv_chart_series_t * ser = lv_chart_get_series_next(obj, NULL);
        lv_rect_rect_dsc_t * rect_dsc = (lv_rect_rect_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);
        uint32_t cnt = lv_chart_get_point_count(obj);

        /*Make older value more transparent/
        rect_dsc->bg_opa = (lv_opa_t)((LV_OPA_COVER * base_dsc->id2) / (cnt - 1));

        /*Make smaller values blue, higher values red/
        int32_t * x_array = lv_chart_get_series_x_array(obj, ser);
        int32_t * y_array = lv_chart_get_series_y_array(obj, ser);
        /*dsc->id is the tells drawing order, but we need the ID of the point being drawn.*/
        uint32_t start_point = lv_chart_get_x_start_point(obj, ser);
        uint32_t p_act = (start_point + base_dsc->id2) % cnt; /*Consider start point to get the index of the array*/
        lv_opa_t x_opa = (lv_opa_t)((x_array[p_act] * LV_OPA_50) / 200);
        lv_opa_t y_opa = (lv_opa_t)((y_array[p_act] * LV_OPA_50) / 1000);

        rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_RED),

```

(continues on next page)

(continua dalla pagina precedente)

```

        lv_palette_main(LV_PALETTE_BLUE),
        x_opa + y_opa);
    }

static void add_data(lv_timer_t * timer)
{
    lv_obj_t * chart = (lv_obj_t *)lv_timer_get_user_data(timer);
    lv_chart_set_next_value2(chart, lv_chart_get_series_next(chart, NULL), (int32_t)lv_rand(0, 200),
                            (int32_t)lv_rand(0, 1000));
}

/***
 * A scatter chart
 */
void lv_example_chart_7(void)
{
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(chart, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_obj_set_style_line_width(chart, 0, LV_PART_ITEMS); /*Remove the lines*/

    lv_chart_set_type(chart, LV_CHART_TYPE_SCATTER);

    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_X, 0, 200);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 1000);

    lv_chart_set_point_count(chart, 50);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 50; i++) {
        lv_chart_set_next_value2(chart, ser, (int32_t)lv_rand(0, 200), (int32_t)lv_rand(0, 1000));
    }

    lv_timer_create(add_data, 100, chart);
}
#endif

```

## Grafico a linee circolari con spazio

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES

static void add_data(lv_timer_t * t)
{
    lv_obj_t * chart = (lv_obj_t *)lv_timer_get_user_data(t);
    lv_chart_series_t * ser = lv_chart_get_series_next(chart, NULL);

    lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));

    uint32_t p = lv_chart_get_point_count(chart);
    uint32_t s = lv_chart_get_x_start_point(chart, ser);
    int32_t * a = lv_chart_get_series_y_array(chart, ser);

    a[(s + 1) % p] = LV_CHART_POINT_NONE;
    a[(s + 2) % p] = LV_CHART_POINT_NONE;
    a[(s + 2) % p] = LV_CHART_POINT_NONE;

    lv_chart_refresh(chart);
}

/***
 * Circular line chart with gap
 */
void lv_example_chart_8(void)
{
    /*Create a stacked_area_chart.obj*/
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_chart_set_update_mode(chart, LV_CHART_UPDATE_MODE_CIRCULAR);
    lv_obj_set_style(chart, 0, 0, LV_PART_INDICATOR);
    lv_obj_set_size(chart, 280, 150);
    lv_obj_center(chart);

    lv_chart_set_point_count(chart, 80);
    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    /*Prefill with data*/
    uint32_t i;
    for(i = 0; i < 80; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
    }

    lv_timer_create(add_data, 300, chart);
}
#endif

```

## 3.7.10 Checkbox

### Checkbox Semplici

```

#include "../../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);

```

(continues on next page)

(continua dalla pagina precedente)

```

if(code == LV_EVENT_VALUE_CHANGED) {
    LV_UNUSED(obj);
    const char * txt = lv_checkbox_get_text(obj);
    const char * state = lv_obj_get_state(obj) & LV_STATE_CHECKED ? "Checked" : "Unchecked";
    LV_UNUSED(txt);
    LV_UNUSED(state);
    LV_LOG_USER("%s: %s", txt, state);
}

void lv_example_checkbox_1(void)
{
    lv_obj_set.flex_flow(lv_screen_active(), LV_FLEX_FLOW_COLUMN);
    lv_obj.set.flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * cb;
    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Apple");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Banana");
    lv_obj.add_state(cb, LV_STATE_CHECKED);
    lv_obj.add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Lemon");
    lv_obj.add_state(cb, LV_STATE_DISABLED);
    lv_obj.add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_obj.add_state(cb, LV_STATE_CHECKED);
    lv_obj.add_state(cb, LV_STATE_DISABLED);
    lv_checkbox.set.text(cb, "Melon\nand a new line");
    lv_obj.add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    lv_obj_update_layout(cb);
}

#endif

```

## Checkboxes come come radio button

```

#include "../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static lv_style_t style_radio;
static lv_style_t style_radio_chk;
static int32_t active_index_1 = 0;
static int32_t active_index_2 = 0;

static void radio_event_handler(lv_event_t * e)
{
    int32_t * active_id = (int32_t *)lv_event_get_user_data(e);
    lv_obj_t * cont = (lv_obj_t *)lv_event_get_current_target(e);
    lv_obj_t * act_cb = lv_event_get_target_obj(e);
    lv_obj_t * old_cb = lv_obj_get_child(cont, *active_id);

    /*Do nothing if the container was clicked*/
    if(act_cb == cont) return;

    lv_obj_remove_state(old_cb, LV_STATE_CHECKED); /*Uncheck the previous radio button*/
    lv_obj_add_state(act_cb, LV_STATE_CHECKED); /*Check the current radio button*/

    *active_id = lv_obj_get_index(act_cb);

    LV_LOG_USER("Selected radio buttons: %d, %d", (int)active_index_1, (int)active_index_2);
}

static void radiobutton_create(lv_obj_t * parent, const char * txt)
{
    lv_obj_t * obj = lv_checkbox_create(parent);
    lv_checkbox.set.text(obj, txt);
    lv_obj.add_flag(obj, LV_OBJ_FLAG_EVENT_BUBBLE);
    lv_obj.add_style(obj, &style_radio, LV_PART_INDICATOR);
    lv_obj.add_style(obj, &style_radio_chk, LV_PART_INDICATOR | LV_STATE_CHECKED);
}

/**
 * Checkboxes as radio buttons
 */
void lv_example_checkbox_2(void)
{
    /* The idea is to enable `LV_OBJ_FLAG_EVENT_BUBBLE` on checkboxes and process the
     * `LV_EVENT_CLICKED` on the container.
     * A variable is passed as event user data where the index of the active
     * radiobutton is saved */

    lv_style_init(&style_radio);
    lv_style_set_radius(&style_radio, LV_RADIUS_CIRCLE);

    lv_style_init(&style_radio_chk);
    lv_style_set_bg_image_src(&style_radio_chk, NULL);

    uint32_t i;
    char buf[32];

    lv_obj_t * cont1 = lv_obj_create(lv_screen_active());
    lv_obj.set.flex_flow(cont1, LV_FLEX_FLOW_COLUMN);
    lv_obj.set.size(cont1, lv_pct(40), lv_pct(80));
    lv_obj.add_event_cb(cont1, radio_event_handler, LV_EVENT_CLICKED, &active_index_1);

    for(i = 0; i < 5; i++) {
        lv_snprintf(buf, sizeof(buf), "A %d", (int)i + 1);
        radiobutton_create(cont1, buf);
    }

    /*Make the first checkbox checked*/
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_add_state(lv_obj_get_child(cont1, 0), LV_STATE_CHECKED);

lv_obj_t * cont2 = lv_obj_create(lv_screen_active());
lv_obj_set_flex_flow(cont2, LV_FLEX_FLOW_COLUMN);
lv_obj_set_size(cont2, lv_pct(40), lv_pct(80));
lv_obj_set_x(cont2, lv_pct(50));
lv_obj_add_event_cb(cont2, radio_event_handler, LV_EVENT_CLICKED, &active_index_2);

for(i = 0; i < 3; i++) {
    lv_snprintf(buf, sizeof(buf), "B %d", (int)i + 1);
    radiobutton_create(cont2, buf);
}

/*Make the first checkbox checked*/
lv_obj_add_state(lv_obj_get_child(cont2, 0), LV_STATE_CHECKED);
}

#endif

```

### 3.7.11 Menù a discesa

#### Menù a discesa semplice

```

#include "../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_dropdown_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Option: %s", buf);
    }
}

void lv_example_dropdown_1(void)
{
    /*Create a normal drop down list/
    lv_obj_t * dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options(dd, "Apple\n"
                           "Banana\n"
                           "Orange\n"
                           "Cherry\n"
                           "Grape\n"
                           "Raspberry\n"
                           "Melon\n"
                           "Orange\n"
                           "Lemon\n"
                           "Nuts");

    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 20);
    lv_obj_add_event_cb(dd, event_handler, LV_EVENT_ALL, NULL);
}

#endif

```

#### Menù a discesa nelle quattro direzioni

```

#include "../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

/***
 * Create a drop down, up, left and right menus
 */
void lv_example_dropdown_2(void)
{
    static const char * opts = "Apple\n"
                             "Banana\n"
                             "Orange\n"
                             "Melon";

    lv_obj_t * dd;
    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 10);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_BOTTOM);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_UP);
    lv_obj_align(dd, LV_ALIGN_BOTTOM_MID, 0, -10);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_RIGHT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_RIGHT);
    lv_obj_align(dd, LV_ALIGN_LEFT_MID, 10, 0);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_LEFT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_LEFT);
    lv_obj_align(dd, LV_ALIGN_RIGHT_MID, -10, 0);
}

#endif

```

## Menù

```
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * dropdown = lv_event_get_target_obj(e);
    char buf[64];
    lv_dropdown_get_selected_str(dropdown, buf, sizeof(buf));
    LV_LOG_USER("%s is selected", buf);
}

/**
 * Create a menu from a drop-down list and show some drop-down list features and styling
 */
void lv_example_dropdown_3(void)
{
    /*Create a drop down list*/
    lv_obj_t * dropdown = lv_dropdown_create(lv_screen_active());
    lv_obj_align(dropdown, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_dropdown_set_options(dropdown, "New project\n"
                            "New file\n"
                            "Save\n"
                            "Save as ...\n"
                            "Open project\n"
                            "Recent projects\n"
                            "Preferences\n"
                            "Exit");

    /*Set a fixed text to display on the button of the drop-down list*/
    lv_dropdown_set_text(dropdown, "Menu");

    /*Use a custom image as down icon and flip it when the list is opened*/
    LV_IMAGE_DECLARE(img_caret_down);
    lv_dropdown_set_symbol(dropdown, &img_caret_down);
    lv_obj_set_style_transform_rotation(dropdown, 1800, LV_PART_INDICATOR | LV_STATE_CHECKED);

    /*In a menu we don't need to show the last clicked item*/
    lv_dropdown_set_selected_highlight(dropdown, false);

    lv_obj_add_event_cb(dropdown, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
}
#endif
```

### 3.7.12 Immagine

#### Immagine da variabile e simbolo

```
#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES

void lv_example_image_1(void)
{
    LV_IMAGE_DECLARE(img_cogwheel_argb);
    lv_obj_t * img1 = lv_image_create(lv_screen_active());
    lv_image_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_CENTER, 0, 0);

    lv_obj_t * img2 = lv_image_create(lv_screen_active());
    lv_image_set_src(img2, LV_SYMBOL_OK "Accept");
    lv_obj_align_to(img2, img1, LV_ALIGN_OUT_BOTTOM_MID, 0, 20);
}

#endif
```

#### Ricolorazione dell'immagine

```
#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_USE_SLIDER && LV_BUILD_EXAMPLES

static lv_obj_t * create_slider(lv_color_t color);
static void slider_event_cb(lv_event_t * e);

static lv_obj_t * red_slider, * green_slider, * blue_slider, * intense_slider;
static lv_obj_t * img1;

/**
 * Demonstrate runtime image re-coloring
 */
void lv_example_image_2(void)
{
    /*Create 4 sliders to adjust RGB color and re-color intensity*/
    red_slider = create_slider(lv_palette_main(LV_PALETTE_RED));
    green_slider = create_slider(lv_palette_main(LV_PALETTE_GREEN));
    blue_slider = create_slider(lv_palette_main(LV_PALETTE_BLUE));
    intense_slider = create_slider(lv_palette_main(LV_PALETTE_GREY));

    lv_slider_set_value(red_slider, LV_OPA_20, LV_ANIM_OFF);
    lv_slider_set_value(green_slider, LV_OPA_90, LV_ANIM_OFF);
    lv_slider_set_value(blue_slider, LV_OPA_60, LV_ANIM_OFF);
    lv_slider_set_value(intense_slider, LV_OPA_50, LV_ANIM_OFF);

    lv_obj_align(red_slider, LV_ALIGN_LEFT_MID, 25, 0);
    lv_obj_align_to(green_slider, red_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(blue_slider, green_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(intense_slider, blue_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);

    /*Now create the actual image*/
    LV_IMAGE_DECLARE(img_cogwheel_argb);
    img1 = lv_image_create(lv_screen_active());
    lv_image_set_src(img1, &img_cogwheel_argb);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_align(img1, LV_ALIGN_RIGHT_MID, -20, 0);
lv_obj_send_event(intense_slider, LV_EVENT_VALUE_CHANGED, NULL);
}

static void slider_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    /*Recolor the image based on the sliders' values*/
    lv_color_t color = lv_color_make((uint8_t)lv_slider_get_value(red_slider),
                                    (uint8_t)lv_slider_get_value(green_slider),
                                    (uint8_t)lv_slider_get_value(blue_slider));
    lv_opa_t intense = (lv_opa_t)lv_slider_get_value(intense_slider);
    lv_obj_set_style_image_recolor_opa(img1, intense, 0);
    lv_obj_set_style_image_recolor(img1, color, 0);
}

static lv_obj_t * create_slider(lv_color_t color)
{
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_slider_set_range(slider, 0, 255);
    lv_obj_set_size(slider, 10, 200);
    lv_obj_set_style_bg_color(slider, color, LV_PART_KNOB);
    lv_obj_set_style_bg_color(slider, lv_color_darken(color, LV_OPA_40), LV_PART_INDICATOR);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    return slider;
}

#endif

```

## Rotazione e zoom

```

#include "../../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES

static void set_angle(void * img, int32_t v)
{
    lv_image_set_rotation((lv_obj_t *)img, v);
}

static void set_scale(void * img, int32_t v)
{
    lv_image_set_scale((lv_obj_t *)img, v);
}

/**
 * Show transformations (zoom and rotation) using a pivot point.
 */
void lv_example_image_3(void)
{
    LV_IMAGE_DECLARE(img_cogwheel_argb);

    /*Now create the actual image*/
    lv_obj_t * img = lv_image_create(lv_screen_active());
    lv_image_set_src(img, &img_cogwheel_argb);
    lv_obj_align(img, LV_ALIGN_CENTER, 50, 50);
    lv_image_set_pivot(img, 0, 0); /*Rotate around the top left corner*/

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, img);
    lv_anim_set_exec_cb(&a, set_angle);
    lv_anim_set_values(&a, 0, 3600);
    lv_anim_set_duration(&a, 5000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, set_scale);
    lv_anim_set_values(&a, 128, 256);
    lv_anim_set_reverse_duration(&a, 3000);
    lv_anim_start(&a);
}

#endif

```

## Offset e stile dell'immagine

```

#include "../../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES

static void ofs_y_anim(void * img, int32_t v)
{
    lv_image_set_offset_y((lv_obj_t *)img, v);
}

/**
 * Image styling and offset
 */
void lv_example_image_4(void)
{
    LV_IMAGE_DECLARE(img_skew_strip);

    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_image_recolor_opa(&style, LV_OPA_COVER);
    lv_style_set_image_recolor(&style, lv_color_black());

    lv_obj_t * img = lv_image_create(lv_screen_active());
    lv_obj_add_style(img, &style, 0);
    lv_image_set_src(img, &img_skew_strip);
    lv_obj_set_size(img, 150, 100);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_center(img);

lv_anim_t a;
lv_anim_init(&a);
lv_anim_set_var(&a, img);
lv_anim_set_exec_cb(&a, ofs_y_anim);
lv_anim_set_values(&a, 0, 100);
lv_anim_set_duration(&a, 3000);
lv_anim_set_reverse_duration(&a, 500);
lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
lv_anim_start(&a);
}

#endif

```

### 3.7.13 Pulsante Immagine

#### Pulsante immagine semplice

```

#include "../../lv_examples.h"
#if LV_USE_IMAGEBUTTON && LV_BUILD_EXAMPLES

void lv_example_imagebutton_1(void)
{
    LV_IMAGE_DECLARE(imagebutton_left);
    LV_IMAGE_DECLARE(imagebutton_right);
    LV_IMAGE_DECLARE(imagebutton_mid);

    /*Create a transition animation on width transformation and recolor.*/
    static lv_style_prop_t tr_prop[] = {LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_IMAGE_RECOLOR_OPA, 0};
    static lv_style_transition_dsc_t tr;
    lv_style_transition_dsc_init(&tr, tr_prop, lv_anim_path_linear, 200, 0, NULL);

    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_text_color(&style_def, lv_color_white());
    lv_style_set_transition(&style_def, &tr);

    /*Darken the button when pressed and make it wider*/
    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_image_recolor_opa(&style_pr, LV_OPA_30);
    lv_style_set_image_recolor(&style_pr, lv_color_black());
    lv_style_set_transform_width(&style_pr, 20);

    /*Create an image button*/
    lv_obj_t * imagebutton1 = lv_imagebutton_create(lv_screen_active());
    lv_imagebutton_set_src(imagebutton1, LV_IMAGEBUTTON_STATE_RELEASED, &imagebutton_left, &imagebutton_mid,
                          &imagebutton_right);
    lv_obj_add_style(imagebutton1, &style_def, 0);
    lv_obj_add_style(imagebutton1, &style_pr, LV_STATE_PRESSED);

    lv_obj_set_width(imagebutton1, 100);
    lv_obj_align(imagebutton1, LV_ALIGN_CENTER, 0, 0);

    /*Create a label on the image button*/
    lv_obj_t * label = lv_label_create(imagebutton1);
    lv_label_set_text(label, "Button");
    lv_obj_align(label, LV_ALIGN_CENTER, 0, -4);
}
#endif

```

### 3.7.14 Tastiera

#### Tastiera con area di testo

```

#include "../../lv_examples.h"
#if LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target_obj(e);
    lv_obj_t * kb = (lv_obj_t *)lv_event_get_user_data(e);
    if(code == LV_EVENT_FOCUSED) {
        lv_keyboard_set_textarea(kb, ta);
        lv_obj_remove_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }

    if(code == LV_EVENT_DEFOCUSSED) {
        lv_keyboard_set_textarea(kb, NULL);
        lv_obj_add_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }
}

void lv_example_keyboard_1(void)
{
    /*Create a keyboard to use it with an of the text areas*/
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());

    /*Create a text area. The keyboard will write here*/
    lv_obj_t * tal;
    tal = lv_textarea_create(lv_screen_active());
    lv_obj_align(tal, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_obj_add_event_cb(tal, ta_event_cb, LV_EVENT_ALL, kb);
    lv_textarea_set_placeholder_text(tal, "Hello");
    lv_obj_set_size(tal, 140, 80);
    lv_obj_t * ta2;
}

```

(continues on next page)

(continua dalla pagina precedente)

```

ta2 = lv_textarea_create(lv_screen_active());
lv_obj_align(ta2, LV_ALIGN_TOP_RIGHT, -10, 10);
lv_obj_add_event_cb(ta2, ta_event_cb, LV_EVENT_ALL, kb);
lv_obj_set_size(ta2, 140, 80);

lv_keyboard_set_textarea(kb, ta1);

/*The keyboard will show Arabic characters if they are enabled */
#if LV_USE_ARABIC_PERSIAN_CHARS && LV_FONT_DEJAVU_16_PERSIAN_HEBREW
    lv_obj_set_style_text_font(kb, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_style_text_font(ta1, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_style_text_font(ta2, &lv_font_dejavu_16_persian_hebrew, 0);
#endif
}
#endif

```

## Tastiera con mappa personalizzata

```

#include "../../../lv_examples.h"
#if LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

void lv_example_keyboard_2(void)
{
    /*Create an AZERTY keyboard map*/
    static const char * kb_map[] = {"A", "Z", "E", "R", "T", "Y", "U", "I", "O", "P", LV_SYMBOL_BACKSPACE, "\n",
                                    "Q", "S", "D", "F", "G", "J", "K", "L", "M", LV_SYMBOL_NEW_LINE, "\n",
                                    "W", "X", "C", "V", "B", "N", ",", ".", ";", "!", "?", "\n",
                                    LV_SYMBOL_CLOSE, " ", " ", " ", LV_SYMBOL_OK, NULL};

    /*Set the relative width of the buttons and other controls*/
    static const lv_buttonmatrix_ctrl_t kb_ctrl[] = {LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4,
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4,
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_6,
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4,
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_2, (lv_buttonmatrix_ctrl_t)(LV_BUTTONMATRIX_CTRL_HIDDEN | LV_BUTTONMATRIX_CTRL_HIDDEN),
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_2, LV_BUTTONMATRIX_CTRL_WIDTH_6, (lv_buttonmatrix_ctrl_t)(LV_BUTTONMATRIX_CTRL_HIDDEN | LV_BUTTONMATRIX_CTRL_HIDDEN),
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_2
    };

    /*Create a keyboard and add the new map as USER_1 mode*/
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());

    lv_keyboard_set_map(kb, LV_KEYBOARD_MODE_USER_1, kb_map, kb_ctrl);
    lv_keyboard_set_mode(kb, LV_KEYBOARD_MODE_USER_1);

    /*Create a text area. The keyboard will write here*/
    lv_obj_t * ta;
    ta = lv_textarea_create(lv_screen_active());
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, 10);
    lv_obj_set_size(ta, lv_pct(90), 80);
    lv_obj_add_state(ta, LV_STATE_FOCUSED);

    lv_keyboard_set_textarea(kb, ta);
}
#endif

```

## Tastiera con disegno

```

#include "../../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_dsc(draw_task);
    bool pressed = false;
    if(lv_keyboard_get_selected_button(obj) == base_dsc->id1 && lv_obj_has_state(obj, LV_STATE_PRESSED)) {
        pressed = true;
    }

    /*When the keyboard draws the buttons...*/
    if(base_dsc->part == LV_PART_ITEMS) {
        /*Get a color based on the button's index*/
        lv_palette_t palette = (lv_palette_t)(base_dsc->id1 % LV_PALETTE_LAST);
        lv_draw_fill_dsc_t * fill_dsc = lv_draw_task_get_fill_dsc(draw_task);
        if(fill_draw_dsc) {
            fill_draw_dsc->color = pressed ? lv_palette_darken(palette, 3) : lv_palette_main(palette);
        }
        lv_draw_label_dsc_t * label_dsc = lv_draw_task_get_label_dsc(draw_task);
        if(label_dsc) {
            /*For the OK symbol, draw a star instead*/
            if(lv_strcmp(label_dsc->text, LV_SYMBOL_OK) == 0) {
                label_dsc->opa = 0; /*Hide the label*/
                LV_IMAGE_DECLARE(img_star);
                lv_image_header_t header;
                lv_result_t res = lv_image_decoder_get_info(&img_star, &header);
                if(res != LV_RESULT_OK) return;

                lv_area_t a;
                lv_area_set(&a, 0, 0, header.w - 1, header.h - 1);
                lv_area_t draw_task_area;
                lv_draw_task_get_area(draw_task, &draw_task_area);
                lv_area_align(&draw_task_area, &a, LV_ALIGN_CENTER, 0, 0);

                lv_draw_image_dsc_t img_draw_dsc;
                lv_draw_image_dsc_init(&img_draw_dsc);
            }
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

        img_draw_dsc.src = &img_star;
        lv_draw_image(base_dsc->layer, &img_draw_dsc, &a);
    }
    else {
        /*For the other labels just pick an lighter color*/
        label_draw_dsc->color = lv_palette_lighten(palette, 4);
    }
}
}

/**
 * Add custom drawer to the keyboard to customize the buttons one by one
 */
void lv_example_keyboard_3(void)
{
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());
    lv_obj_add_event_cb(kb, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(kb, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_obj_center(kb);
}
#endif

```

### 3.7.15 Etichetta

#### A capo automatico, ricolorazione e scorrimento

```

#include "../../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/***
 * Show line wrap, re-color, line align and text scrolling.
 */
void lv_example_label_1(void)
{
    lv_obj_t * label1 = lv_label_create(lv_screen_active());
    lv_label_set_long_mode(label1, LV_LABEL_LONG_MODE_WRAP);      /*Break the long lines*/
    lv_label_set_recolor(label1, true);                            /*Enable re-coloring by commands in the text*/
    lv_label_set_text(label1, "#0000ff Re-color# #ff00ff words# #ff0000 of # label, align the lines to the center "
                           "and wrap long text automatically.");
    lv_obj_set_width(label1, 150); /*Set smaller width to make the lines wrap*/
    lv_obj_set_style_text_align(label1, LV_TEXT_ALIGN_CENTER, 0);
    lv_obj_align(label1, LV_ALIGN_CENTER, 0, -40);

    lv_obj_t * label2 = lv_label_create(lv_screen_active());
    lv_label_set_long_mode(label2, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR); /*Circular scroll*/
    lv_obj_set_width(label2, 150);
    lv_label_set_text(label2, "It is a circularly scrolling text. ");
    lv_obj_align(label2, LV_ALIGN_CENTER, 0, 40);
}

#endif

```

#### Ombreggiatura del testo

```

#include "../../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/***
 * Create a fake text shadow
 */
void lv_example_label_2(void)
{
    /*Create a style for the shadow*/
    static lv_style_t style_shadow;
    lv_style_init(&style_shadow);
    lv_style_set_text_opa(&style_shadow, LV_OPA_30);
    lv_style_set_text_color(&style_shadow, lv_color_black());

    /*Create a label for the shadow first (it's in the background)*/
    lv_obj_t * shadow_label = lv_label_create(lv_screen_active());
    lv_obj_add_style(shadow_label, &style_shadow, 0);

    /*Create the main label*/
    lv_obj_t * main_label = lv_label_create(lv_screen_active());
    lv_label_set_text(main_label, "A simple method to create\n"
                           "shadows on a text.\n"
                           "It even works with\n"
                           "newlines and spaces.");
    /*Set the same text for the shadow label*/
    lv_label_set_text(shadow_label, lv_label_get_text(main_label));

    /*Position the main label*/
    lv_obj_align(main_label, LV_ALIGN_CENTER, 0, 0);

    /*Shift the second label down and to the right by 2 pixel*/
    lv_obj_align_to(shadow_label, main_label, LV_ALIGN_TOP_LEFT, 2, 2);
}

#endif

```

## Visualizzazione di testi LTR, RTL e in cinese

```

#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES && LV_FONT_DEJAVU_16_PERSIAN_HEBREW && LV_FONT_SOURCE_HAN_SANS_SC_16_CJK && LV_USE_BIDI

/***
 * Show mixed LTR, RTL and Chinese label
 */
void lv_example_label_3(void)
{
    lv_obj_t * ltr_label = lv_label_create(lv_screen_active());
    lv_label_set_text(ltr_label, "In modern terminology, a microcontroller is similar to a system on a chip (SoC).");
    lv_obj_set_style_text_font(ltr_label, &lv_font_montserrat_16, 0);
    lv_obj_set_width(ltr_label, 310);
    lv_obj_align(ltr_label, LV_ALIGN_TOP_LEFT, 5, 5);

    lv_obj_t * rtl_label = lv_label_create(lv_screen_active());
    lv_label_set_text(rtl_label,
                      "处理器，即微控制器（CPU - Central Processing Unit）.");
    lv_obj_set_style_base_dir(rtl_label, LV_BASE_DIR_RTL, 0);
    lv_obj_set_style_text_font(rtl_label, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_width(rtl_label, 310);
    lv_obj_align(rtl_label, LV_ALIGN_LEFT_MID, 5, 0);

    lv_obj_t * cz_label = lv_label_create(lv_screen_active());
    lv_label_set_text(cz_label,
                      "嵌入式系统\nEmbedded System");
    lv_obj_set_style_text_font(cz_label, &lv_font_source_han_sans_sc_16_cjk, 0);
    lv_obj_set_width(cz_label, 310);
    lv_obj_align(cz_label, LV_ALIGN_BOTTOM_LEFT, 5, -5);
}

#endif

```

## Disegno di etichette con gradiente di colore

```

#include "../../lv_examples.h"

#if LV_USE_LABEL && LV_FONT_MONTserrat_24 && LV_USE_CANVAS && LV_BUILD_EXAMPLES && LV_DRAW_SW_COMPLEX

#define MASK_WIDTH 150
#define MASK_HEIGHT 60

static void generate_mask(lv_draw_buf_t * mask, int32_t w, int32_t h, const char * txt)
{
    /*Create a "8 bit alpha" canvas and clear it*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, mask);
    lv_canvas_fill_bg(canvas, lv_color_black(), LV_OPA_TRANSP);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    /*Draw a label to the canvas. The result "image" will be used as mask*/
    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_color_white();
    label_dsc.align = LV_TEXT_ALIGN_CENTER;
    label_dsc.text = txt;
    label_dsc.font = &lv_font_montserrat_24;
    lv_area_t a = {0, 0, -w - 1, h - 1};
    lv_draw_label(&layer, &label_dsc, &a);

    lv_canvas_finish_layer(canvas, &layer);
    lv_obj_delete(canvas);
}

/***
 * Draw label with gradient color
 */
void lv_example_label_4(void)
{
    /* Create the mask of a text by drawing it to a canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(mask, MASK_WIDTH, MASK_HEIGHT, LV_COLOR_FORMAT_L8);
    LV_DRAW_BUF_INIT_STATIC(mask);

    generate_mask(&mask, MASK_WIDTH, MASK_HEIGHT, "Text with gradient");

    /* Create an object from where the text will be masked out.
     * Now it's a rectangle with a gradient but it could be an image too*/
    lv_obj_t * grad = lv_obj_create(lv_screen_active());
    lv_obj_set_size(grad, MASK_WIDTH, MASK_HEIGHT);
    lv_obj_center(grad);
    lv_obj_set_style_bg_color(grad, lv_color_hex(0xff0000), 0);
    lv_obj_set_style_bg_grad_color(grad, lv_color_hex(0x0000ff), 0);
    lv_obj_set_style_bg_grad_dir(grad, LV_GRAD_DIR_HOR, 0);
    lv_obj_set_style_bitmap_src(grad, &mask, 0);
}

#endif

```

## Personalizzazione dell'animazione a scorrimento circolare

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/** 
 * Show customizing the circular scrolling animation of a label with `LV_LABEL_LONG_MODE_SCROLL_CIRCULAR`
 * long mode.
 */
void lv_example_label_5(void)
{
    static lv_anim_t animation_template;
    static lv_style_t label_style;
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_init(&animation_template);
lv_anim_set_delay(&animation_template, 1000);           /*Wait 1 second to start the first scroll*/
lv_anim_set_repeat_delay(&animation_template,
                        3000);   /*Repeat the scroll 3 seconds after the label scrolls back to the initial position*/
lv_anim_set_repeat_count(&animation_template, LV_ANIM_REPEAT_INFINITE);

/*Initialize the label style with the animation template*/
lv_style_init(&label_style);
lv_style_set_anim(&label_style, &animation_template);

lv_obj_t * label1 = lv_label_create(lv_screen_active());
lv_label_set_long_mode(label1, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);      /*Circular scroll*/
lv_obj_set_width(label1, 150);
lv_label_set_text(label1, "It is a circularly scrolling text. ");
lv_obj_align(label1, LV_ALIGN_CENTER, 0, 40);
lv_obj_add_style(label1, &label_style, LV_STATE_DEFAULT);           /*Add the style to the label*/
}

#endif

```

## Carattere a spaziatura fissa

```

#include "../../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES && LV_FONT_MONTserrat_20
static bool fix_w_get_glyph_dsc(const lv_font_t * font, lv_font_glyph_dsc_t * dsc, uint32_t letter,
                                uint32_t letter_next)
{
    bool ret = lv_font_get_glyph_dsc_fmt_txt(font, dsc, letter, letter_next);
    if(!ret) return false;

    /* Set a fixed width */
    dsc->adv_w = 20;
    dsc->ofs_x = (dsc->adv_w - dsc->box_w) / 2;
    return true;
}

void lv_example_label_6(void)
{
    /* Clone the original font and override its behavior */
    static lv_font_t mono_font;
    mono_font = lv_font_montserrat_20;
    mono_font.get_glyph_dsc = fix_w_get_glyph_dsc;

    /* Create a label with normal font */
    lv_obj_t * label1 = lv_label_create(lv_screen_active());
    lv_obj_set_style_text_font(label1, &lv_font_montserrat_20, 0);
    lv_label_set_text(label1, "0123.Wabc");

    /* Create a label with fixed-width glyph descriptor override */
    lv_obj_t * label2 = lv_label_create(lv_screen_active());
    lv_obj_set_y(label2, 30);
    lv_obj_set_style_text_font(label2, &mono_font, 0);
    lv_label_set_text(label2, "0123.Wabc");
}
#endif

```

## 3.7.16 LED

### LED con stile personalizzato

```

#include "../../../lv_examples.h"
#if LV_USE_LED && LV_BUILD_EXAMPLES

/***
 * Create LED's with different brightness and color
 */
void lv_example_led_1(void)
{
    /*Create a LED and switch it OFF*/
    lv_obj_t * led1 = lv_led_create(lv_screen_active());
    lv_obj_align(led1, LV_ALIGN_CENTER, -80, 0);
    lv_led_off(led1);

    /*Copy the previous LED and set a brightness*/
    lv_obj_t * led2 = lv_led_create(lv_screen_active());
    lv_obj_align(led2, LV_ALIGN_CENTER, 0, 0);
    lv_led_set_brightness(led2, 150);
    lv_led_set_color(led2, lv_palette_main(LV_PALETTE_RED));

    /*Copy the previous LED and switch it ON*/
    lv_obj_t * led3 = lv_led_create(lv_screen_active());
    lv_obj_align(led3, LV_ALIGN_CENTER, 80, 0);
    lv_led_on(led3);
}

#endif

```

## 3.7.17 Line

### Linea semplice

```

#include "../../../lv_examples.h"
#if LV_USE_LINE && LV_BUILD_EXAMPLES

```

(continues on next page)

(continua dalla pagina precedente)

```

void lv_example_line_1(void)
{
    /*Create an array for the points of the line*/
    static lv_point_precise_t line_points[] = { {5, 5}, {70, 70}, {120, 10}, {180, 60}, {240, 10} };

    /*Create style*/
    static lv_style_t style_line;
    lv_style_init(&style_line);
    lv_style_set_line_width(&style_line, 8);
    lv_style_set_line_color(&style_line, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_line_rounded(&style_line, true);

    /*Create a line and apply the new style*/
    lv_obj_t * linel;
    linel = lv_line_create(lv_screen_active());
    lv_line_set_points(linel, line_points, 5);      /*Set the points*/
    lv_obj_add_style(linel, &style_line, 0);
    lv_obj_center(linel);
}

#endif

```

### 3.7.18 Lista

#### Lista semplice

```

#include "../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES
static lv_obj_t * list1;

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        LV_UNUSED(obj);
        LV_LOG_USER("Clicked: %s", lv_list_get_button_text(list1, obj));
    }
}
void lv_example_list_1(void)
{
    /*Create a list*/
    list1 = lv_list_create(lv_screen_active());
    lv_obj_set_size(list1, 180, 220);
    lv_obj_center(list1);

    /*Add buttons to the list*/
    lv_obj_t * btn;
    lv_list_add_text(list1, "File");
    btn = lv_list_add_button(list1, LV_SYMBOL_FILE, "New");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_DIRECTORY, "Open");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_SAVE, "Save");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_CLOSE, "Delete");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_EDIT, "Edit");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Connectivity");
    btn = lv_list_add_button(list1, LV_SYMBOL_BLUETOOTH, "Bluetooth");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_GPS, "Navigation");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_USB, "USB");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_BATTERY_FULL, "Battery");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Exit");
    btn = lv_list_add_button(list1, LV_SYMBOL_OK, "Apply");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_CLOSE, "Close");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
}
#endif

```

#### Ordinamento di un elenco utilizzando i pulsanti Su e Giù

```

#include "../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES
static lv_obj_t * list1;
static lv_obj_t * list2;

static lv_obj_t * currentButton = NULL;

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked: %s", lv_list_get_button_text(list1, obj));

        if(currentButton == obj) {
            currentButton = NULL;
        } else {
            currentButton = obj;
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * parent = lv_obj_get_parent(obj);
int32_t i;
for(i = 0; i < (int32_t)lv_obj_get_child_count(parent); i++) {
    lv_obj_t * child = lv_obj_get_child(parent, i);
    if(child == currentButton) {
        lv_obj_add_state(child, LV_STATE_CHECKED);
    } else {
        lv_obj_remove_state(child, LV_STATE_CHECKED);
    }
}
}

static void event_handler_top(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) ||
       (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;
        lv_obj_move_background(currentButton);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_up(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;
        int32_t index = lv_obj_get_index(currentButton);
        if(index <= 0) return;
        lv_obj_move_to_index(currentButton, index - 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_center(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;

        lv_obj_t * parent = lv_obj_get_parent(currentButton);
        const int32_t pos = (int32_t)lv_obj_get_child_count(parent) / 2;

        lv_obj_move_to_index(currentButton, pos);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_dn(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;
        const int32_t index = lv_obj_get_index(currentButton);

        lv_obj_move_to_index(currentButton, index + 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_bottom(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_CLICKED) {
        if(currentButton == NULL) return;
        lv_obj_move_foreground(currentButton);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_swap(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        uint32_t cnt = lv_obj_get_child_count(list1);
        for(int i = 0; i < 100; i++) {
            lv_obj_t * obj = lv_obj_get_child(list1, (int32_t)lv_rand(0, cnt - 1));
            lv_obj_move_to_index(obj, (int32_t)lv_rand(0, cnt - 1));
            if(currentButton != NULL) {
                lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
            }
        }
    }
}

void lv_example_list_2(void)
{
    /*Create a list*/
    list1 = lv_list_create(lv_screen_active());
    lv_obj_set_size(list1, lv_pct(60), lv_pct(100));
    lv_obj_set_style_pad_row(list1, 5, 0);

    /*Add buttons to the list*/
    lv_obj_t * btn;
    int i;
    for(i = 0; i < 15; i++) {
        btn = lv_button_create(list1);
        lv_obj_set_width(btn, lv_pct(50));
        lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

        lv_obj_t * lab = lv_label_create(btn);
        lv_label_set_text_fmt(lab, "Item %d", i);
    }

    /*Select the first button by default*/
    currentButton = lv_obj_get_child(list1, 0);
    lv_obj_add_state(currentButton, LV_STATE_CHECKED);

    /*Create a second list with up and down buttons*/
}

```

(continues on next page)

(continua dalla pagina precedente)

```

list2 = lv_list_create(lv_screen_active());
lv_obj_set_size(list2, lv_pct(40), lv_pct(100));
lv_obj_align(list2, LV_ALIGN_TOP_RIGHT, 0, 0);
lv_obj_set_flex_flow(list2, LV_FLEX_FLOW_COLUMN);

btn = lv_list_add_button(list2, NULL, "Top");
lv_obj_add_event_cb(btn, event_handler_top, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_UP, "Up");
lv_obj_add_event_cb(btn, event_handler_up, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_LEFT, "Center");
lv_obj_add_event_cb(btn, event_handler_center, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_DOWN, "Down");
lv_obj_add_event_cb(btn, event_handler_dn, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, NULL, "Bottom");
lv_obj_add_event_cb(btn, event_handler_bottom, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_SHUFFLE, "Shuffle");
lv_obj_add_event_cb(btn, event_handler_swap, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);
}

#endif

```

### 3.7.19 Lottie

#### Caricamento di un'animazione Lottie da un array

```

#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_LOTTIE

/** 
 * Load an lottie animation from data
 */
void lv_example_lottie_1(void)
{
    extern const uint8_t lv_example_lottie_approve[];
    extern const size_t lv_example_lottie_approve_size;

    lv_obj_t * lottie = lv_lottie_create(lv_screen_active());
    lv_lottie_set_src_data(lottie, lv_example_lottie_approve, lv_example_lottie_approve_size);

#if LV_DRAW_BUF_ALIGN == 4 && LV_DRAW_BUF_STRIDE_ALIGN == 1
    /*If there are no special requirements, just declare a buffer
     x4 because the Lottie is rendered in ARGB8888_PREMULTIPLIED format*/
    static uint8_t buf[64 * 64 * 4];
    lv_lottie_set_buffer(lottie, 64, 64, buf);
#else
    /*For GPUs and special alignment/stride setting use a draw_buf instead*/
    LV_DRAW_BUF_DEFINE(draw_buf, 64, 64, LV_COLOR_FORMAT_ARGB8888_PREMULTIPLIED);
    lv_lottie_set_draw_buf(lottie, &draw_buf);
#endif

    lv_obj_center(lottie);
}

#else

void lv_example_lottie_1(void)
{
    /*fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Lottie cannot be previewed online");
    lv_obj_center(label);
}

#endif /*LV_USE_LOTTIE*/
#endif /*LV_BUILD_EXAMPLES*/

```

#### Caricamento di un'animazione Lottie da un file

```

#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_LOTTIE

/** 
 * Load an lottie animation from file
 */
void lv_example_lottie_2(void)
{
    lv_obj_t * lottie = lv_lottie_create(lv_screen_active());
    lv_lottie_set_src_file(lottie, "lvgl/examples/widgets/lottie/lv_example_lottie_approve.json");

#if LV_DRAW_BUF_ALIGN == 4 && LV_DRAW_BUF_STRIDE_ALIGN == 1
    /*If there are no special requirements, just declare a buffer
     x4 because the Lottie is rendered in ARGB8888_PREMULTIPLIED format*/
    static uint8_t buf[64 * 64 * 4];
    lv_lottie_set_buffer(lottie, 64, 64, buf);
#else
    /*For GPUs and special alignment/stride setting use a draw_buf instead*/

```

(continues on next page)

(continua dalla pagina precedente)

```

LV_DRAW_BUF_DEFINE(draw_buf, 64, 64, LV_COLOR_FORMAT_ARGB8888_PREMULTIPLIED);
lv_lottie_set_draw_buf(lottie, &draw_buf);
#endif

lv_obj_center(lottie);
}

#else

void lv_example_lottie_2(void)
{
    /*fallback for online examples*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Lottie cannot be previewed online");
    lv_obj_center(label);
}

#endif /*LV_USE_LOTTIE*/
#endif /*LV_BUILD_EXAMPLES*/

```

## 3.7.20 Menù

### Menù semplice

```

#include "../../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

void lv_example_menu_1(void)
{
    /*Create a menu object*/
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);
}

#endif

```

### Menù semplice con pulsante principale

```

#include "../../lv_examples.h"
#if LV_USE_MENU && LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void back_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);

    if(lv_menu_back_button_is_root(menu, obj)) {
        lv_obj_t * mbox1 = lv_msgbox_create(NULL);
        lv_msgbox_add_title(mbox1, "Hello");
        lv_msgbox_add_text(mbox1, "Root back btn click.");
        lv_msgbox_add_close_button(mbox1);
    }
}

void lv_example_menu_2(void)
{
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_menu_set_mode_root_back_button(menu, LV_MENU_ROOT_BACK_BUTTON_ENABLED);
    lv_obj_add_event_cb(menu, back_event_handler, LV_EVENT_CLICKED, menu);
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/*Create a main page*/
lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

cont = lv_menu_cont_create(main_page);
label = lv_label_create(cont);
lv_label_set_text(label, "Item 1");

cont = lv_menu_cont_create(main_page);
label = lv_label_create(cont);
lv_label_set_text(label, "Item 2");

cont = lv_menu_cont_create(main_page);
label = lv_label_create(cont);
lv_label_set_text(label, "Item 3 (Click me!)");
lv_menu_set_load_page_event(menu, cont, sub_page);

lv_menu_set_page(menu, main_page);
}

#endif

```

## Menù semplice con intestazione personalizzata

```

#include "../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

void lv_example_menu_3(void)
{
    /*Create a menu object*/
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    /*Modify the header*/
    lv_obj_t * back_btn = lv_menu_get_main_header_back_button(menu);
    lv_obj_t * back_button_label = lv_label_create(back_btn);
    lv_label_set_text(back_button_label, "Back");

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create sub pages*/
    lv_obj_t * sub_1_page = lv_menu_page_create(menu, "Page 1");

    cont = lv_menu_cont_create(sub_1_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    lv_obj_t * sub_2_page = lv_menu_page_create(menu, "Page 2");

    cont = lv_menu_cont_create(sub_2_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    lv_obj_t * sub_3_page = lv_menu_page_create(menu, "Page 3");

    cont = lv_menu_cont_create(sub_3_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_1_page);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_2_page);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_3_page);

    lv_menu_set_page(menu, main_page);
}

#endif

```

## Menù semplice con pulsante mobile per aggiungere una nuova pagina di menu

```

#include "../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

static uint32_t btn_cnt = 1;
static lv_obj_t * main_page;
static lv_obj_t * menu;

static void float_button_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    btn_cnt++;

    lv_obj_t * cont;
    lv_obj_t * label;

    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

```

(continues on next page)

(continua dalla pagina precedente)

```

cont = lv_menu_cont_create(sub_page);
label = lv_label_create(cont);
lv_label_set_text_fmt(label, "Hello, I am hiding inside %" LV_PRIu32 "", btn_cnt);

cont = lv_menu_cont_create(main_page);
label = lv_label_create(cont);
lv_label_set_text_fmt(label, "Item %" LV_PRIu32 "", btn_cnt);
lv_menu_set_load_page_event(menu, cont, sub_page);

lv_obj_scroll_to_view_recursive(cont, LV_ANIM_ON);
}

void lv_example_menu_4(void)
{
    /*Create a menu object*/
    menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding inside the first item");

    /*Create a main page*/
    main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);

    /*Create floating btn*/
    lv_obj_t * float_btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(float_btn, 50, 50);
    lv_obj_add_flag(float_btn, LV_OBJ_FLAG_FLOATING);
    lv_obj_align(float_btn, LV_ALIGN_BOTTOM_RIGHT, -10, -10);
    lv_obj_add_event_cb(float_btn, float_button_event_cb, LV_EVENT_CLICKED, menu);
    lv_obj_set_radius(float_btn, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_image_src(float_btn, LV_SYMBOL_PLUS, 0);
    lv_obj_set_style_text_font(float_btn, lv_theme_get_font_large(float_btn), 0);
}

#endif

```

## Menù complesso

```

#include "../../../lv_examples.h"
#if LV_USE_MENU && LV_USE_MSGBOX && LV_BUILD_EXAMPLES

typedef enum {
    LV_MENU_ITEM_BUILDER_VARIANT_1,
    LV_MENU_ITEM_BUILDER_VARIANT_2
} lv_menu_builder_variant_t;

static void back_event_handler(lv_event_t * e);
static void switch_handler(lv_event_t * e);
static lv_obj_t * root_page;
static lv_obj_t * create_text(lv_obj_t * parent, const char * icon, const char * txt,
                             lv_menu_builder_variant_t builder_variant);
static lv_obj_t * create_slider(lv_obj_t * parent,
                               const char * icon, const char * txt, int32_t min, int32_t max, int32_t val);
static lv_obj_t * create_switch(lv_obj_t * parent,
                               const char * icon, const char * txt, bool chk);

void lv_example_menu_5(void)
{
    lv_obj_t * menu = lv_menu_create(lv_screen_active());

    lv_color_t bg_color = lv_obj_get_style_bg_color(menu, LV_PART_MAIN);
    if(lv_color_brightness(bg_color) > 127) {
        lv_obj_set_style_bg_color(menu, lv_color_darken(lv_obj_get_style_bg_color(menu, LV_PART_MAIN), 10), 0);
    } else {
        lv_obj_set_style_bg_color(menu, lv_color_darken(lv_obj_get_style_bg_color(menu, LV_PART_MAIN), 50), 0);
    }
    lv_menu_set_mode_root_back_button(menu, LV_MENU_ROOT_BACK_BUTTON_ENABLED);
    lv_obj_add_event_cb(menu, back_event_handler, LV_EVENT_CLICKED, menu);
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * section;

    /*Create sub pages*/
    lv_obj_t * sub_mechanics_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_mechanics_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    lv_menu_separator_create(sub_mechanics_page);
    section = lv_menu_section_create(sub_mechanics_page);
    create_slider(section, LV_SYMBOL_SETTINGS, "Velocity", 0, 150, 120);
    create_slider(section, LV_SYMBOL_SETTINGS, "Acceleration", 0, 150, 50);
    create_slider(section, LV_SYMBOL_SETTINGS, "Weight limit", 0, 150, 80);

    lv_obj_t * sub_sound_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_sound_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    lv_menu_separator_create(sub_sound_page);
    section = lv_menu_section_create(sub_sound_page);
    create_switch(section, LV_SYMBOL_AUDIO, "Sound", false);

    lv_obj_t * sub_display_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_display_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_menu_separator_create(sub_display_page);
section = lv_menu_section_create(sub_display_page);
create_slider(section, LV_SYMBOL_SETTINGS, "Brightness", 0, 150, 100);

lv_obj_t * sub_software_info_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_software_info_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN),
                        0);
section = lv_menu_section_create(sub_software_info_page);
create_text(section, NULL, "Version 1.0", LV_MENU_ITEM_BUILDER_VARIANT_1);

lv_obj_t * sub_legal_info_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_legal_info_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN),
                        0);
section = lv_menu_section_create(sub_legal_info_page);
for(uint32_t i = 0; i < 15; i++) {
    create_text(section, NULL,
                "This is a long long long long long long long long text, if it is long enough it may scroll.",
                LV_MENU_ITEM_BUILDER_VARIANT_1);
}

lv_obj_t * sub_about_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_about_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
lv_menu_separator_create(sub_about_page);
section = lv_menu_section_create(sub_about_page);
cont = create_text(section, NULL, "Software information", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_software_info_page);
cont = create_text(section, NULL, "Legal information", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_legal_info_page);

lv_obj_t * sub_menu_mode_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_menu_mode_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
lv_menu_separator_create(sub_menu_mode_page);
section = lv_menu_section_create(sub_menu_mode_page);
cont = create_switch(section, LV_SYMBOL_AUDIO, "Sidebar enable", true);
lv_obj_add_event_cb(lv_obj_get_child(cont, 2), switch_handler, LV_EVENT_VALUE_CHANGED, menu);

/*Create a root page*/
root_page = lv_menu_page_create(menu, "Settings");
lv_obj_set_style_pad_hor(root_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
section = lv_menu_section_create(root_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Mechanics", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_mechanics_page);
cont = create_text(section, LV_SYMBOL_AUDIO, "Sound", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_sound_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Display", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_display_page);

create_text(root_page, NULL, "Others", LV_MENU_ITEM_BUILDER_VARIANT_1);
section = lv_menu_section_create(root_page);
cont = create_text(section, NULL, "About", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_about_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Menu mode", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_menu_mode_page);

lv_menu_set_sidebar_page(menu, root_page);

lv_obj_send_event(lv_obj_get_child(lv_obj_get_child(lv_menu_get_cur_sidebar_page(menu), 0), 0), LV_EVENT_CLICKED,
                  NULL);
}

static void back_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);

    if(lv_menu_back_button_is_root(menu, obj)) {
        lv_obj_t * mbox1 = lv_msgbox_create(NULL);
        lv_msgbox_add_title(mbox1, "Hello");
        lv_msgbox_add_text(mbox1, "Root back btn click.");
        lv_msgbox_add_close_button(mbox1);
    }
}

static void switch_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        if(lv_obj_has_state(obj, LV_STATE_CHECKED)) {
            lv_menu_set_page(menu, NULL);
            lv_menu_set_sidebar_page(menu, root_page);
            lv_obj_send_event(lv_obj_get_child(lv_obj_get_child(lv_menu_get_cur_sidebar_page(menu), 0), 0), LV_EVENT_CLICKED,
                              NULL);
        }
        else {
            lv_menu_set_sidebar_page(menu, NULL);
            lv_menu_clear_history(menu); /* Clear history because we will be showing the root page later */
            lv_menu_set_page(menu, root_page);
        }
    }
}

static lv_obj_t * create_text(lv_obj_t * parent, const char * icon, const char * txt,
                            lv_menu_builder_variant_t builder_variant)
{
    lv_obj_t * obj = lv_menu_cont_create(parent);

    lv_obj_t * img = NULL;
    lv_obj_t * label = NULL;

    if(icon) {
        img = lv_image_create(obj);
        lv_image_set_src(img, icon);
    }

    if(txt) {
        label = lv_label_create(obj);
        lv_label_set_text(label, txt);
        lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
        lv_obj_set_flex_grow(label, 1);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

if(builder_variant == LV_MENU_ITEM_BUILDER_VARIANT_2 && icon && txt) {
    lv_obj_add_flag(img, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
    lv_obj_swap(img, label);
}

return obj;
}

static lv_obj_t * create_slider(lv_obj_t * parent, const char * icon, const char * txt, int32_t min, int32_t max,
                               int32_t val)
{
    lv_obj_t * obj = create_text(parent, icon, txt, LV_MENU_ITEM_BUILDER_VARIANT_2);

    lv_obj_t * slider = lv_slider_create(obj);
    lv_obj_set_flex_grow(slider, 1);
    lv_slider_set_range(slider, min, max);
    lv_slider_set_value(slider, val, LV_ANIM_OFF);

    if(icon == NULL) {
        lv_obj_add_flag(slider, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
    }

    return obj;
}

static lv_obj_t * create_switch(lv_obj_t * parent, const char * icon, const char * txt, bool chk)
{
    lv_obj_t * obj = create_text(parent, icon, txt, LV_MENU_ITEM_BUILDER_VARIANT_1);

    lv_obj_t * sw = lv_switch_create(obj);
    lv_obj_add_state(sw, chk ? LV_STATE_CHECKED : LV_STATE_DEFAULT);

    return obj;
}

#endif

```

### 3.7.21 Casella Messaggi

#### Casella messaggi semplice

```

#include "../../lv_examples.h"
#if LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    LV_UNUSED(label);
    LV_LOG_USER("Button %s clicked", lv_label_get_text(label));
}

void lv_example_msgbox_1(void)
{
    lv_obj_t * mbox1 = lv_msgbox_create(NULL);

    lv_msgbox_add_title(mbox1, "Hello");

    lv_msgbox_add_text(mbox1, "This is a message box with two buttons.");
    lv_msgbox_add_close_button(mbox1);

    lv_obj_t * btn;
    btn = lv_msgbox_add_footer_button(mbox1, "Apply");
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);
    btn = lv_msgbox.add_footer_button(mbox1, "Cancel");
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);
    return;
}
#endif

```

#### Casella messaggi con scorrimento e stile

```

#include "../../lv_examples.h"
#if LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void minimize_button_event_cb(lv_event_t * e)
{
    lv_obj_t * mbox = (lv_obj_t *) lv_event_get_user_data(e);
    lv_obj_add_flag(mbox, LV_OBJ_FLAG_HIDDEN);
}

void lv_example_msgbox_2(void)
{
    lv_obj_t * setting = lv_msgbox_create(lv_screen_active());
    lv_obj_set_style_clip_corner(setting, true, 0);

    /* setting fixed size */
    lv_obj_set_size(setting, 300, 200);

    /* setting's titlebar/header */
    lv_msgbox_add_title(setting, "Setting");
    lv_obj_t * minimize_button = lv_msgbox.add_header_button(setting, LV_SYMBOL_MINUS);
    lv_obj_add_event_cb(minimize_button, minimize_button_event_cb, LV_EVENT_CLICKED, setting);
    lv_msgbox.add_close_button(setting);

    /* setting's content*/
    lv_obj_t * content = lv_msgbox.get_content(setting);
    lv_obj_set_flex_flow(content, LV_FLEX_FLOW_COLUMN);
    lv_obj.set_flex_align(content, LV_ALIGN_START, LV_ALIGN_CENTER, LV_ALIGN_CENTER);
    lv_obj.set_style_pad_right(content, -1, LV_PART_SCROLLBAR);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * cont_brightness = lv_obj_create(content);
lv_obj_set_size(cont_brightness, lv_pct(100), LV_SIZE_CONTENT);
lv_obj_set_flex_flow(cont_brightness, LV_FLEX_FLOW_COLUMN);
lv_obj_set_flex_align(cont_brightness, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER);

lv_obj_t * lb_brightness = lv_label_create(cont_brightness);
lv_label_set_text(lb_brightness, "Brightness : ");
lv_obj_t * slider_brightness = lv_slider_create(cont_brightness);
lv_obj_set_width(slider_brightness, lv_pct(100));
lv_slider_set_value(slider_brightness, 50, LV_ANIM_OFF);

lv_obj_t * cont_speed = lv_obj_create(content);
lv_obj_set_size(cont_speed, lv_pct(100), LV_SIZE_CONTENT);
lv_obj_set_flex_flow(cont_speed, LV_FLEX_FLOW_COLUMN);
lv_obj_set_flex_align(cont_speed, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER);

lv_obj_t * lb_speed = lv_label_create(cont_speed);
lv_label_set_text(lb_speed, "Speed : ");
lv_obj_t * slider_speed = lv_slider_create(cont_speed);
lv_obj_set_width(slider_speed, lv_pct(100));
lv_slider_set_value(slider_speed, 80, LV_ANIM_OFF);

/* footer */
lv_obj_t * apply_button = lv_msgbox_add_footer_button(setting, "Apply");
lv_obj_set_flex_grow(apply_button, 1);

lv_obj_t * cancel_button = lv_msgbox_add_footer_button(setting, "Cancel");
lv_obj_set_flex_grow(cancel_button, 1);

lv_obj_t * footer = lv_msgbox_get_footer(setting);
lv_obj_set_style_bg_color(footer, lv_palette_main(LV_PALETTE_INDIGO), 0);
lv_obj_set_style_bg_opa(footer, LV_OPA_100, 0);
}
#endif

```

## 3.7.22 Roller

### Roller semplice

```

#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected month: %s\n", buf);
    }
}

/**
 * An infinite roller with the name of the months
 */
void lv_example_roller_1(void)
{
    lv_obj_t * roller1 = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller1,
        "January\n"
        "February\n"
        "March\n"
        "April\n"
        "May\n"
        "June\n"
        "July\n"
        "August\n"
        "September\n"
        "October\n"
        "November\n"
        "December",
        LV_ROLLER_MODE_INFINITE);

    lv_roller_set_visible_row_count(roller1, 4);
    lv_obj_center(roller1);
    lv_obj_add_event_cb(roller1, event_handler, LV_EVENT_ALL, NULL);
}
#endif

```

### Roller con stili

```

#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_FONT_MONTserrat_22 && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected value: %s", buf);
    }
}

/**
 * Roller with various alignments and larger text in the selected area
 */
void lv_example_roller_2(void)

```

(continues on next page)

(continua dalla pagina precedente)

```
{
    /*A style to make the selected option larger*/
    static lv_style_t style_sel;
    lv_style_init(&style_sel);
    lv_style_set_text_font(&style_sel, &lv_font_montserrat_22);
    lv_style_set_bg_color(&style_sel, lv_color_hex3(0xf88));
    lv_style_set_border_width(&style_sel, 2);
    lv_style_set_border_color(&style_sel, lv_color_hex3(0xf00));

    const char * opts = "1\n2\n3\n4\n5\n6\n7\n8\n9\n10";
    lv_obj_t * roller;

    /*A roller on the left with left aligned text, and custom width*/
    roller = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 2);
    lv_obj_set_width(roller, 100);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_LEFT, 0);
    lv_obj_set_style_bg_color(roller, lv_color_hex3(0x0f0), 0);
    lv_obj_set_style_bg_grad_color(roller, lv_color_hex3(0xaafa), 0);
    lv_obj_set_style_bg_grad_dir(roller, LV_GRAD_DIR_VER, 0);
    lv_obj_align(roller, LV_ALIGN_LEFT_MID, 10, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 2, LV_ANIM_OFF);

    /*A roller on the middle with center aligned text, and auto (default) width*/
    roller = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 3);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_align(roller, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 5, LV_ANIM_OFF);

    /*A roller on the right with right aligned text, and custom width*/
    roller = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 4);
    lv_obj_set_width(roller, 80);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_RIGHT, 0);
    lv_obj_align(roller, LV_ALIGN_RIGHT_MID, -10, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 8, LV_ANIM_OFF);
}

#endif
```

## Aggiungere una maschera di dissolvenza al roller

```
#include "../../../lv_examples.h"

#if LV_USE_ROLLER && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES && 1

static void generate_mask(lv_draw_buf_t * mask)
{
    /*Create a "8 bit alpha" canvas and clear it*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, mask);
    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_TRANSP);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    /*Draw a label to the canvas. The result "image" will be used as mask*/
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.bg_grad.dir = LV_GRAD_DIR_VER;
    rect_dsc.bg_grad.stops[0].color = lv_color_black();
    rect_dsc.bg_grad.stops[1].color = lv_color_white();
    rect_dsc.bg_grad.stops[0].opa = LV_OPA_COVER;
    rect_dsc.bg_grad.stops[1].opa = LV_OPA_COVER;
    lv_area_t a = {0, 0, mask->header.w - 1, mask->header.h / 2 - 10};
    lv_draw_rect(&layer, &rect_dsc, &a);

    a.y1 = mask->header.h / 2 + 10;
    a.y2 = mask->header.h - 1;
    rect_dsc.bg_grad.stops[0].color = lv_color_white();
    rect_dsc.bg_grad.stops[1].color = lv_color_black();
    lv_draw_rect(&layer, &rect_dsc, &a);

    lv_canvas_finish_layer(canvas, &layer);

    /*Comment it to make the mask visible*/
    lv_obj_delete(canvas);
}

/**
 * Add a fade mask to roller.
 */
void lv_example_roller_3(void)
{
    lv_obj_set_style_bg_color(lv_screen_active(), lv_palette_main(LV_PALETTE_BLUE_GREY), 0);

    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_color_black());
    lv_style_set_text_color(&style, lv_color_white());
    lv_style_set_border_width(&style, 0);
    lv_style_set_radius(&style, 0);

    lv_obj_t * roller1 = lv_roller_create(lv_screen_active());
    lv_obj_add_style(roller1, &style, 0);
    lv_obj_set_style_bg_opa(roller1, LV_OPA_50, LV_PART_SELECTED);

    lv_roller_set_options(roller1,
        "January\n"
```

(continues on next page)

(continua dalla pagina precedente)

```

"February\n"
"March\n"
"April\n"
"May\n"
"June\n"
"July\n"
"August\n"
"September\n"
"October\n"
"November\n"
"December",
LV_ROLLER_MODE_NORMAL);

lv_obj_center(roller1);
lv_roller_set_visible_row_count(roller1, 4);

/* Create the mask to make the top and bottom part of roller faded.
 * The width and height are empirical values for simplicity/
LV_DRAW_BUF_DEFINE_STATIC(mask, 130, 150, LV_COLOR_FORMAT_L8);
LV_DRAW_BUF_INIT_STATIC(mask);

generate_mask(&mask);
lv_obj_set_style_bitmap_mask_src(roller1, &mask, 0);
}
#endif

```

### 3.7.23 Scala

#### Una semplice scala orizzontale

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/**
 * A simple horizontal scale
 */
void lv_example_scale_1(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, lv_pct(80), 100);
    lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 31);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 10, 40);
}

#endif

```

#### Una scala verticale con sezioni e stile personalizzato

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/**
 * An vertical scale with section and custom styling
 */
void lv_example_scale_2(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 60, 200);
    lv_scale_set_label_show(scale, true);
    lv_scale_set_mode(scale, LV_SCALE_MODE_VERTICAL_RIGHT);
    lv_obj_center(scale);

    lv_scale_set_total_tick_count(scale, 21);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_scale_set_range(scale, 0, 100);

    static const char * custom_labels[] = {"0 °C", "25 °C", "50 °C", "75 °C", "100 °C", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);

    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));

    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
    lv_style_set_width(&indicator_style, 10U); /*Tick length*/
    lv_style_set_line_width(&indicator_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_palette_lighten(LV_PALETTE_BLUE, 2));
    lv_style_set_width(&minor_ticks_style, 5U); /*Tick length*/
    lv_style_set_line_width(&minor_ticks_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

    static lv_style_t main_line_style;

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_init(&main_line_style);
/* Main line properties */
lv_style_set_line_color(&main_line_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
lv_style_set_line_width(&main_line_style, 2U); /* Tick width
lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

/* Add a section */
static lv_style_t section_minor_tick_style;
static lv_style_t section_label_style;
static lv_style_t section_main_line_style;

lv_style_init(&section_label_style);
lv_style_init(&section_minor_tick_style);
lv_style_init(&section_main_line_style);

/* Label style properties */
lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
lv_style_set_text_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));

lv_style_set_line_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_line_width(&section_label_style, 5U); /*Tick width*/

lv_style_set_line_color(&section_minor_tick_style, lv_palette_lighten(LV_PALETTE_RED, 2));
lv_style_set_line_width(&section_minor_tick_style, 4U); /*Tick width*/

/* Main line properties */
lv_style_set_line_color(&section_main_line_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_line_width(&section_main_line_style, 4U); /*Tick width*/

/* Configure section styles */
lv_scale_section_t * section = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section, 75, 100);
lv_scale_set_section_style_indicator(scale, section, &section_label_style);
lv_scale_set_section_style_items(scale, section, &section_minor_tick_style);
lv_scale_set_section_style_main(scale, section, &section_main_line_style);

lv_obj_set_style_bg_color(scale, lv_palette_main(LV_PALETTE_BLUE_GREY), 0);
lv_obj_set_style_bg_opa(scale, LV_OPA_50, 0);
lv_obj_set_style_pad_left(scale, 8, 0);
lv_obj_set_style_radius(scale, 8, 0);
lv_obj_set_style_pad_ver(scale, 20, 0);
}

#endif

```

## Una semplice scala circolare

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

LV_IMAGE_DECLARE(img_hand);

static lv_obj_t * needle_line;
static lv_obj_t * needle_img;

static void set_needle_line_value(void * obj, int32_t v)
{
    lv_scale_set_line_needle_value((lv_obj_t *)obj, needle_line, 60, v);
}

static void set_needle_img_value(void * obj, int32_t v)
{
    lv_scale_set_image_needle_value((lv_obj_t *)obj, needle_img, v);
}

/**
 * A simple round scale
 */
void lv_example_scale_3(void)
{
    lv_obj_t * scale_line = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale_line, 150, 150);
    lv_scale_set_mode(scale_line, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale_line, LV_OPA_COVER, 0);
    lv_obj_set_style_bg_color(scale_line, lv_palette_lighten(LV_PALETTE_RED, 5), 0);
    lv_obj_set_style_radius(scale_line, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(scale_line, true, 0);
    lv_obj_align(scale_line, LV_ALIGN_LEFT_MID, LV_PCT(2), 0);

    lv_scale_set_label_show(scale_line, true);

    lv_scale_set_total_tick_count(scale_line, 31);
    lv_scale_set_major_tick_every(scale_line, 5);

    lv_obj_set_style_length(scale_line, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale_line, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale_line, 10, 40);

    lv_scale_set_angle_range(scale_line, 270);
    lv_scale_set_rotation(scale_line, 135);

    needle_line = lv_line_create(scale_line);

    lv_obj_set_style_line_width(needle_line, 6, LV_PART_MAIN);
    lv_obj_set_style_line_rounded(needle_line, true, LV_PART_MAIN);

    lv_anim_t anim_scale_line;
    lv_anim_init(&anim_scale_line);
    lv_anim_set_var(&anim_scale_line, scale_line);
    lv_anim_set_exec_cb(&anim_scale_line, set_needle_line_value);
    lv_anim_set_duration(&anim_scale_line, 1000);
    lv_anim_set_repeat_count(&anim_scale_line, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_reverse_duration(&anim_scale_line, 1000);
    lv_anim_set_values(&anim_scale_line, 10, 40);
    lv_anim_start(&anim_scale_line);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * scale_img = lv_scale_create(lv_screen_active());

lv_obj_set_size(scale_img, 150, 150);
lv_scale_set_mode(scale_img, LV_SCALE_MODE_ROUND_INNER);
lv_obj_set_style_bg_opa(scale_img, LV_OPA_COVER, 0);
lv_obj_set_style_bg_color(scale_img, lv_palette_lighten(LV_PALETTE_RED, 5), 0);
lv_obj_set_style_radius(scale_img, LV_RADIUS_CIRCLE, 0);
lv_obj_set_style_clip_corner(scale_img, true, 0);
lv_obj_align(scale_img, LV_ALIGN_RIGHT_MID, LV_PCT(-2), 0);

lv_scale_set_label_show(scale_img, true);

lv_scale_set_total_tick_count(scale_img, 31);
lv_scale_set_major_tick_every(scale_img, 5);

lv_obj_set_style_length(scale_img, 5, LV_PART_ITEMS);
lv_obj_set_style_length(scale_img, 10, LV_PART_INDICATOR);
lv_scale_set_range(scale_img, 10, 40);

lv_scale_set_angle_range(scale_img, 270);
lv_scale_set_rotation(scale_img, 135);

/* image must point to the right. E.g. -0----->*
needle_img = lv_image_create(scale_img);
lv_image_set_src(needle_img, &img_hand);
lv_obj_align(needle_img, LV_ALIGN_CENTER, 47, -2);
lv_image_set_pivot(needle_img, 3, 4);

lv_anim_t anim_scale_img;
lv_anim_init(&anim_scale_img);
lv_anim_set_var(&anim_scale_img, scale_img);
lv_anim_set_exec_cb(&anim_scale_img, set_needle_img_value);
lv_anim_set_duration(&anim_scale_img, 1000);
lv_anim_set_repeat_count(&anim_scale_img, LV_ANIM_REPEAT_INFINITE);
lv_anim_set_reverse_duration(&anim_scale_img, 1000);
lv_anim_set_values(&anim_scale_img, 10, 40);
lv_anim_start(&anim_scale_img);
}

#endif

```

## Una scala circolare con sezione e stile personalizzato

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A round scale with section and custom styling
*/
void lv_example_scale_4(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 150, 150);
    lv_scale_set_label_show(scale, true);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_OUTER);
    lv_obj_center(scale);

    lv_scale_set_total_tick_count(scale, 21);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 0, 100);

    static const char * custom_labels[] = {"0 °C", "25 °C", "50 °C", "75 °C", "100 °C", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);

    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));

    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
    lv_style_set_width(&indicator_style, 10U); /*Tick length*/
    lv_style_set_line_width(&indicator_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_palette_lighten(LV_PALETTE_BLUE, 2));
    lv_style_set_width(&minor_ticks_style, 5U); /*Tick length*/
    lv_style_set_line_width(&minor_ticks_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

    static lv_style_t main_line_style;
    lv_style_init(&main_line_style);
    /* Main line properties */
    lv_style_set_arc_color(&main_line_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
    lv_style_set_arc_width(&main_line_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

    /* Add a section */
    static lv_style_t section_minor_tick_style;
    static lv_style_t section_label_style;
    static lv_style_t section_main_line_style;

    lv_style_init(&section_label_style);
    lv_style_init(&section_minor_tick_style);
    lv_style_init(&section_main_line_style);

    /* Label style properties */
    lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));
    lv_style_set_line_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_line_width(&section_label_style, 5U); /*Tick width*/
lv_style_set_line_color(&section_minor_tick_style, lv_palette_lighten(LV_PALETTE_RED, 2));
lv_style_set_line_width(&section_minor_tick_style, 4U); /*Tick width*/

/* Main line properties */
lv_style_set_arc_color(&section_main_line_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_arc_width(&section_main_line_style, 4U); /*Tick width*/

/* Configure section styles */
lv_scale_section_t * section = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section, 75, 100);
lv_scale_set_section_style_indicator(scale, section, &section_label_style);
lv_scale_set_section_style_items(scale, section, &section_minor_tick_style);
lv_scale_set_section_style_main(scale, section, &section_main_line_style);

}

#endif

```

## Una scala con sezione e stile personalizzato

```

#include "../../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/**
 * An scale with section and custom styling
 */
void lv_example_scale_5(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, lv_display_get_horizontal_resolution(NULL) / 2, lv_display_get_vertical_resolution(NULL) / 2);
    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 10);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 25, 35);

    static const char * custom_labels[3] = {"One", "Two", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);
    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_color_hex(0xff00ff));
    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_color_hex(0x00ff00));
    lv_style_set_width(&indicator_style, 10U); // Tick length
    lv_style_set_line_width(&indicator_style, 2U); // Tick width
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_color_hex(0xffff00));
    lv_style_set_width(&minor_ticks_style, 5U); // Tick length
    lv_style_set_line_width(&minor_ticks_style, 2U); // Tick width
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

    static lv_style_t main_line_style;
    lv_style_init(&main_line_style);
    /* Main line properties */
    lv_style_set_line_color(&main_line_style, lv_color_hex(0x0000ff));
    lv_style_set_line_width(&main_line_style, 2U); // Tick width
    lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

    lv_obj_center(scale);

    /* Add a section */
    static lv_style_t section_minor_tick_style;
    static lv_style_t section_label_style;

    lv_style_init(&section_label_style);
    lv_style_init(&section_minor_tick_style);

    /* Label style properties */
    lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&section_label_style, lv_color_hex(0xffffffff));
    lv_style_set_text_letter_space(&section_label_style, 10);
    lv_style_set_text_opa(&section_label_style, LV_OPA_50);

    lv_style_set_line_color(&section_label_style, lv_color_hex(0xffff00));
    // lv_style_set_width(&section_label_style, 20U); // Tick length
    lv_style_set_line_width(&section_label_style, 5U); // Tick width

    lv_style_set_line_color(&section_minor_tick_style, lv_color_hex(0x0000ff));
    // lv_style_set_width(&section_label_style, 20U); // Tick length
    lv_style_set_line_width(&section_minor_tick_style, 4U); // Tick width

    /* Configure section styles */
    lv_scale_section_t * section = lv_scale_add_section(scale);
    lv_scale_set_section_range(scale, section, 25, 30);
    lv_scale_set_section_style_indicator(scale, section, &section_label_style);
    lv_scale_set_section_style_items(scale, section, &section_minor_tick_style);
}

#endif

```

## Una scala circolare con più lancette, simile a un orologio

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

#if LV_USE_FLOAT
#define my_PRIprecise "f"
#else
#define my_PRIprecise LV_PRId32
#endif

static lv_obj_t * scale;
static lv_obj_t * minute_hand;
static lv_obj_t * hour_hand;
static lv_point_precise_t minute_hand_points[2];
static int32_t hour;
static int32_t minute;

static void timer_cb(lv_timer_t * timer)
{
    LV_UNUSED(timer);

    minute++;
    if(minute > 59) {
        minute = 0;
        hour++;
        if(hour > 11) {
            hour = 0;
        }
    }

    /**
     * the scale will store the needle line points in the existing
     * point array if one was set with `lv_line_set_points_mutable`.
     * Otherwise, it will allocate the needle line points.
     */

    /* the scale will store the minute hand line points in `minute_hand_points` */
    lv_scale_set_line_needle_value(scale, minute_hand, 60, minute);
    /* log the points that were stored in the array */
    LV_LOG_USER(
        "minute hand points - "
        "0: (%" PRIprecise ", %" PRIprecise "), "
        "1: (%" PRIprecise ", %" PRIprecise ")",
        minute_hand_points[0].x, minute_hand_points[0].y,
        minute_hand_points[1].x, minute_hand_points[1].y
    );

    /* the scale will allocate the hour hand line points */
    lv_scale_set_line_needle_value(scale, hour_hand, 40, hour * 5 + (minute / 12));
}

/**
 * A round scale with multiple needles, resembling a clock
 */
void lv_example_scale_6(void)
{
    scale = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale, 150, 150);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale, LV_OPA_60, 0);
    lv_obj_set_style_bg_color(scale, lv_color_black(), 0);
    lv_obj_set_style_radius(scale, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(scale, true, 0);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 61);
    lv_scale_set_major_tick_every(scale, 5);

    static const char * hour_ticks[] = {"12", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", NULL};
    lv_scale_set_text_src(scale, hour_ticks);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);

    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_palette_main(LV_PALETTE_YELLOW));

    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_length(&indicator_style, 8); /* tick length */
    lv_style_set_line_width(&indicator_style, 2); /* tick width */
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    /* Minor tick properties */
    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_length(&minor_ticks_style, 6); /* tick length */
    lv_style_set_line_width(&minor_ticks_style, 2); /* tick width */
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

    /* Main line properties */
    static lv_style_t main_line_style;
    lv_style_init(&main_line_style);
    lv_style_set_arc_color(&main_line_style, lv_color_black());
    lv_style_set_arc_width(&main_line_style, 5);
    lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

    lv_scale_set_range(scale, 0, 60);

    lv_scale_set_angle_range(scale, 360);
    lv_scale_set_rotation(scale, 270);

    minute_hand = lv_line_create(scale);
    lv_line_set_points_mutable(minute_hand, minute_hand_points, 2);

    lv_obj_set_style_line_width(minute_hand, 3, 0);
    lv_obj_set_style_line_rounded(minute_hand, true, 0);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_style_line_color(minute_hand, lv_color_white(), 0);
hour_hand = lv_line_create(scale);

lv_obj_set_style_line_width(hour_hand, 5, 0);
lv_obj_set_style_line_rounded(hour_hand, true, 0);
lv_obj_set_style_line_color(hour_hand, lv_palette_main(LV_PALETTE_RED), 0);

hour = 11;
minute = 5;
lv_timer_t * timer = lv_timer_create(timer_cb, 250, NULL);
lv_timer_ready(timer);
}

#endif

```

## Personalizzazione del colore dell'etichetta principale della scala con l'evento LV\_EVENT\_DRAW\_TASK\_ADDED

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

#include "../../../lvgl_private.h" /*To expose the fields of lv_draw_task_t*/

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        if(label_draw_dsc) {
            const lv_color_t color_idx[7] = {
                lv_palette_main(LV_PALETTE_RED),
                lv_palette_main(LV_PALETTE_ORANGE),
                lv_palette_main(LV_PALETTE_YELLOW),
                lv_palette_main(LV_PALETTE_GREEN),
                lv_palette_main(LV_PALETTE_CYAN),
                lv_palette_main(LV_PALETTE_BLUE),
                lv_palette_main(LV_PALETTE_PURPLE),
            };
            uint32_t major_tick = lv_scale_get_major_tick_every(obj);
            label_draw_dsc->color = color_idx[base_dsc->id1 / major_tick];

            /*Free the previously allocated text if needed*/
            if(label_draw_dsc->text_local) lv_free((void *)label_draw_dsc->text);

            /*Malloc the text and set text_local as 1 to make LVGL automatically free the text.
             * (Local texts are malloc'd internally by LVGL. Mimic this behavior here too)*/
            char tmp_buffer[20] = {0}; /* Big enough buffer */
            lv_snprintf(tmp_buffer, sizeof(tmp_buffer), "%lf", (double)base_dsc->id2);
            label_draw_dsc->text = lv_strdup(tmp_buffer);
            label_draw_dsc->text_local = 1;

            lv_text_attributes_t attributes = {0};
            lv_point_t size;

            attributes.text_flags = LV_TEXT_FLAG_NONE;
            attributes.letter_space = 0;
            attributes.line_space = 0;
            attributes.max_width = 1000;

            lv_text_get_size(&size, label_draw_dsc->text, label_draw_dsc->font, &attributes);
            int32_t new_w = size.x;
            int32_t old_w = lv_area_get_width(&draw_task->area);

            /* Distribute the new size equally on both sides */
            draw_task->area.x1 -= (new_w - old_w) / 2;
            draw_task->area.x2 += ((new_w - old_w) + 1) / 2; /* +1 for rounding */
        }
    }

    /**
     * Customizing scale major tick label color with `LV_EVENT_DRAW_TASK_ADDED` event
     */
    void lv_example_scale_7(void)
    {
        lv_obj_t * scale = lv_scale_create(lv_screen_active());
        lv_obj_set_size(scale, lv_pct(80), 100);
        lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
        lv_obj_center(scale);

        lv_scale_set_label_show(scale, true);

        lv_scale_set_total_tick_count(scale, 31);
        lv_scale_set_major_tick_every(scale, 5);

        lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
        lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
        lv_scale_set_range(scale, 10, 40);

        lv_obj_add_event_cb(scale, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
        lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    }
}

```

## Una scala circolare con etichette ruotate e traslate

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A simple round scale with label/tick translation
 */
void lv_example_scale_8(void)
{
    lv_obj_t * scale_line = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale_line, 150, 150);
    lv_scale_set_mode(scale_line, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale_line, LV_OPA_COVER, 0);
    lv_obj_set_style_bg_color(scale_line, lv_palette_lighten(LV_PALETTE_RED, 5), 0);
    lv_obj_set_style_radius(scale_line, LV_RADIUS_CIRCLE, 0);
    lv_obj_align(scale_line, LV_ALIGN_LEFT_MID, LV_PCT(2), 0);

    /*Set the texts' and major ticks' style (make the texts rotated)*/
    lv_obj_set_style_transform_rotation(scale_line, LV_SCALE_LABEL_ROTATE_MATCH_TICKS | LV_SCALE_LABEL_ROTATE_KEEP_UPRIGHT,
                                      LV_PART_INDICATOR);
    lv_obj_set_style_translate_x(scale_line, 10, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale_line, 15, LV_PART_INDICATOR);
    lv_obj_set_style_radial_offset(scale_line, 10, LV_PART_INDICATOR);

    /*Set the style of the minor ticks*/
    lv_obj_set_style_length(scale_line, 10, LV_PART_ITEMS);
    lv_obj_set_style_radial_offset(scale_line, 5, LV_PART_ITEMS);
    lv_obj_set_style_line_opa(scale_line, LV_OPA_50, LV_PART_ITEMS);

    lv_scale_set_label_show(scale_line, true);
    lv_scale_set_total_tick_count(scale_line, 31);
    lv_scale_set_major_tick_every(scale_line, 5);

    lv_scale_set_range(scale_line, 10, 40);
    lv_scale_set_angle_range(scale_line, 270);
    lv_scale_set_rotation(scale_line, 135);

    lv_obj_t * needle_line = lv_line_create(scale_line);

    lv_obj_set_style_line_width(needle_line, 3, LV_PART_MAIN);
    lv_obj_set_style_line_rounded(needle_line, true, LV_PART_MAIN);
    lv_scale_set_line_needle_value(scale_line, needle_line, 60, 33);
}

#endif
```

## Una scala orizzontale con etichette ruotate e traslate

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A simple horizontal scale with transforms
 */
void lv_example_scale_9(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 200, 100);
    lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);
    lv_obj_set_style_transform_rotation(scale, 450, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale, 30, LV_PART_INDICATOR);
    lv_obj_set_style_translate_x(scale, 5, LV_PART_INDICATOR);

    lv_scale_set_total_tick_count(scale, 31);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 10, 40);
}

#endif
```

## Uno stile di scala circolare che simula un cardiofrequenzimetro

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES && LV_FONT_MONTserrat_40 && LV_FONT_MONTserrat_18

static int32_t hr_value = 98;
static int8_t hr_step = 1;
static lv_obj_t * needle_line = NULL;
static lv_obj_t * hr_value_label = NULL;
static lv_obj_t * bpm_label = NULL;
static lv_obj_t * scale = NULL;

typedef struct {
    lv_style_t items;
    lv_style_t indicator;
    lv_style_t main;
} section_styles_t;

static section_styles_t zone1_styles;
static section_styles_t zone2_styles;
static section_styles_t zone3_styles;
static section_styles_t zone4_styles;
static section_styles_t zone5_styles;

static lv_color_t get_hr_zone_color(int32_t hr)
```

(continues on next page)

(continua dalla pagina precedente)

```

{
    if(hr < 117) return lv_palette_main(LV_PALETTE_GREY); /* Zone 1 */
    else if(hr < 135) return lv_palette_main(LV_PALETTE_BLUE); /* Zone 2 */
    else if(hr < 158) return lv_palette_main(LV_PALETTE_GREEN); /* Zone 3 */
    else if(hr < 176) return lv_palette_main(LV_PALETTE_ORANGE); /* Zone 4 */
    else return lv_palette_main(LV_PALETTE_RED); /* Zone 5 */
}

static void hr_anim_timer_cb(lv_timer_t * timer)
{
    LV_UNUSED(timer);

    hr_value += hr_step;

    if(hr_value >= 195) {
        hr_value = 195;
        hr_step = -1;
    }
    else if(hr_value <= 98) {
        hr_value = 98;
        hr_step = 1;
    }

    /* Update needle */
    lv_scale_set_line_needle_value(scale, needle_line, -8, hr_value);

    /* Update HR text */
    lv_label_set_text_fmt(hr_value_label, "%d", hr_value);

    /* Update text color based on zone */
    lv_color_t zone_color = get_hr_zone_color(hr_value);
    lv_obj_set_style_text_color(hr_value_label, zone_color, 0);
    lv_obj_set_style_text_color(bpm_label, zone_color, 0);
}

static void init_section_styles(section_styles_t * styles, lv_color_t color)
{
    lv_style_init(&styles->items);
    lv_style_set_line_color(&styles->items, color);
    lv_style_set_line_width(&styles->items, 0);

    lv_style_init(&styles->indicator);
    lv_style_set_line_color(&styles->indicator, color);
    lv_style_set_line_width(&styles->indicator, 0);

    lv_style_init(&styles->main);
    lv_style_set_arc_color(&styles->main, color);
    lv_style_set_arc_width(&styles->main, 20);
}

static void add_section(lv_obj_t * target_scale,
                      int32_t from,
                      int32_t to,
                      const section_styles_t * styles)
{
    lv_scale_section_t * sec = lv_scale_add_section(target_scale);
    lv_scale_set_section_range(target_scale, sec, from, to);
    lv_scale_set_section_style_items(target_scale, sec, &styles->items);
    lv_scale_set_section_style_indicator(target_scale, sec, &styles->indicator);
    lv_scale_set_section_style_main(target_scale, sec, &styles->main);
}

void lv_example_scale_10(void)
{
    scale = lv_scale_create(lv_screen_active());
    lv_obj_center(scale);
    lv_obj_set_size(scale, 200, 200);

    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_scale_set_range(scale, 98, 195);
    lv_scale_set_total_tick_count(scale, 15);
    lv_scale_set_major_tick_every(scale, 3);
    lv_scale_set_angle_range(scale, 280);
    lv_scale_set_rotation(scale, 130);
    lv_scale_set_label_show(scale, false);

    lv_obj_set_style_length(scale, 6, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_obj_set_style_arc_width(scale, 0, LV_PART_MAIN);

    /* Zone 1: (Grey) */
    init_section_styles(&zone1_styles, lv_palette_main(LV_PALETTE_GREY));
    add_section(scale, 98, 117, &zone1_styles);

    /* Zone 2: (Blue) */
    init_section_styles(&zone2_styles, lv_palette_main(LV_PALETTE_BLUE));
    add_section(scale, 117, 135, &zone2_styles);

    /* Zone 3: (Green) */
    init_section_styles(&zone3_styles, lv_palette_main(LV_PALETTE_GREEN));
    add_section(scale, 135, 158, &zone3_styles);

    /* Zone 4: (Orange) */
    init_section_styles(&zone4_styles, lv_palette_main(LV_PALETTE_ORANGE));
    add_section(scale, 158, 176, &zone4_styles);

    /* Zone 5: (Red) */
    init_section_styles(&zone5_styles, lv_palette_main(LV_PALETTE_RED));
    add_section(scale, 176, 195, &zone5_styles);

    needle_line = lv_line_create(scale);

    /* Optional styling */
    lv_obj_set_style_line_color(needle_line, lv_color_black(), LV_PART_MAIN);
    lv_obj_set_style_line_width(needle_line, 12, LV_PART_MAIN);
    lv_obj_set_style_length(needle_line, 20, LV_PART_MAIN);
    lv_obj_set_style_line_rounded(needle_line, false, LV_PART_MAIN);
    lv_obj_set_style_pad_right(needle_line, 50, LV_PART_MAIN);

    int32_t current_hr = 145;

    lv_scale_set_line_needle_value(scale, needle_line, 0, current_hr);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * circle = lv_obj_create(lv_scr_act());
lv_obj_set_size(circle, 130, 130);
lv_obj_center(circle);

lv_obj_set_style_radius(circle, LV_RADIUS_CIRCLE, 0);

lv_obj_set_style_bg_color(circle, lv_obj_get_style_bg_color(lv_scr_act(), LV_PART_MAIN), 0);
lv_obj_set_style_bg_opa(circle, LV_OPA_COVER, 0);
lv_obj_set_style_border_width(circle, 0, LV_PART_MAIN);

lv_obj_t * hr_container = lv_obj_create(circle);
lv_obj_center(hr_container);
lv_obj_set_size(hr_container, lv_pct(100), LV_SIZE_CONTENT);
lv_obj_set_style_bg_opa(hr_container, LV_OPA_TRANSPI, 0);
lv_obj_set_style_border_width(hr_container, 0, 0);
lv_obj_set_layout(hr_container, LV_LAYOUT_FLEX);
lv_obj_set_flex_flow(hr_container, LV_FLEX_FLOW_COLUMN);
lv_obj_set_style_pad_all(hr_container, 0, LV_PART_MAIN);
lv_obj_set_style_pad_row(hr_container, 0, 0);
lv_obj_set_flex_align(hr_container, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

hr_value_label = lv_label_create(hr_container);
lv_label_set_text_fmt(hr_value_label, "%d", current_hr);
lv_obj_set_style_text_font(hr_value_label, &lv_font_montserrat_40, 0);
lv_obj_set_style_text_align(hr_value_label, LV_TEXT_ALIGN_CENTER, 0);

bpm_label = lv_label_create(hr_container);
lv_label_set_text(bpm_label, "bpm");
lv_obj_set_style_text_font(bpm_label, &lv_font_montserrat_18, 0);
lv_obj_set_style_text_align(bpm_label, LV_TEXT_ALIGN_CENTER, 0);

lv_color_t zone_color = get_hr_zone_color(current_hr);
lv_obj_set_style_text_color(hr_value_label, zone_color, 0);
lv_obj_set_style_text_color(bpm_label, zone_color, 0);

lv_timer_create(hr_anim_timer_cb, 80, NULL);
}

#endif

```

## Uno stile di scala circolare che simula un widget tramonto/alba

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES && LV_FONT_MONTserrat_12 && LV_FONT_MONTserrat_14 && LV_FONT_MONTserrat_16 && LV_FONT_MONTserrat_20

static void label_color_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    if(!draw_task) return;

    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_event_get_draw_dsc(draw_task);
    if(!base_dsc || base_dsc->part != LV_PART_INDICATOR) return;

    lv_draw_label_dsc_t * label_dsc = lv_event_get_label_dsc(draw_task);
    if(!label_dsc || !label_dsc->text) return;

    const char * txt = label_dsc->text;

    if(lv_strcmp(txt, "06") == 0 || lv_strcmp(txt, "12") == 0 ||
       lv_strcmp(txt, "18") == 0 || lv_strcmp(txt, "24") == 0) {
        label_dsc->color = lv_color_white();
    }
    else {
        label_dsc->color = lv_palette_darken(LV_PALETTE_GREY, 1);
    }
}

void lv_example_scale_11(void)
{
    lv_obj_t * bg = lv_obj_create(lv_screen_active());
    lv_obj_set_size(bg, 210, 210);
    lv_obj_center(bg);
    lv_obj_set_style_radius(bg, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_color(bg, lv_palette_darken(LV_PALETTE_GREY, 4), 0);
    lv_obj_set_style_bg_opa(bg, LV_OPA_COVER, 0);
    lv_obj_remove_flag(bg, LV_OBJ_FLAG_SCROLLABLE);
    lv_obj_set_style_pad_all(bg, 0, LV_PART_MAIN);

    lv_obj_t * scale = lv_scale_create(bg);
    lv_obj_center(scale);
    lv_obj_set_size(scale, 150, 150);
    lv_obj_set_style_arc_width(scale, 5, LV_PART_MAIN);

    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_OUTER);
    lv_scale_set_range(scale, 0, 24);
    lv_scale_set_total_tick_count(scale, 25);
    lv_scale_set_major_tick_every(scale, 1);
    lv_scale_set_angle_range(scale, 360);
    lv_scale_set_rotation(scale, 105);
    lv_scale_set_label_show(scale, true);
    lv_obj_set_style_text_font(scale, &lv_font_montserrat_12, LV_PART_INDICATOR);
    lv_obj_set_style_pad_radial(scale, -6, LV_PART_INDICATOR);

    /* Rotate the labels of the ticks */
    lv_obj_set_style_transform_rotation(scale, LV_SCALE_LABEL_ROTATE_MATCH_TICKS | LV_SCALE_LABEL_ROTATE_KEEP_UPRIGHT,
                                      LV_PART_INDICATOR);

    /* Style for major ticks */
    static lv_style_t style_ticks;
    lv_style_init(&style_ticks);
    lv_style_set_line_color(&style_ticks, lv_palette_darken(LV_PALETTE_GREY, 1));
    lv_style_set_line_width(&style_ticks, 2);
    lv_style_set_width(&style_ticks, 10);
    lv_obj_add_style(scale, &style_ticks, LV_PART_INDICATOR);

    /* Style for NIGHT - blue */
    static lv_style_t style_night;

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_init(&style_night);
lv_style_set_arc_color(&style_night, lv_palette_main(LV_PALETTE_BLUE));

/* Style for DAY – dark yellow */
static lv_style_t style_day;
lv_style_init(&style_day);
lv_style_set_arc_color(&style_day, lv_palette_darker(LV_PALETTE_YELLOW, 3));

/* NIGHT section */
lv_scale_section_t * section_night1 = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section_night1, 17, 5);
lv_scale_set_section_style_main(scale, section_night1, &style_night);

/* DAY section */
lv_scale_section_t * section_day = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section_day, 5, 17);
lv_scale_set_section_style_main(scale, section_day, &style_day);

static const char * hour_labels[] = {
    "01", "02", "03", "04", "05",
    "06", "07", "08", "09", "10",
    "11", "12", "13", "14", "15",
    "16", "17", "18", "19", "20",
    "21", "22", "23", "24",
    NULL
};
lv_scale_set_text_src(scale, hour_labels);

lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
lv_obj_add_event_cb(scale, label_color_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);

lv_obj_t * today = lv_label_create(bg);
lv_label_set_text(today, "TODAY");
lv_obj_set_style_text_font(today, &lv_font_montserrat_16, 0);
lv_obj_set_style_text_color(today, lv_color_white(), 0);
lv_obj_align(today, LV_ALIGN_TOP_MID, 0, 60);

lv_obj_t * sunrise_lbl = lv_label_create(bg);
lv_label_set_text(sunrise_lbl, "SUNRISE");
lv_obj_set_style_text_font(sunrise_lbl, &lv_font_montserrat_14, 0);
lv_obj_set_style_text_color(sunrise_lbl, lv_palette_main(LV_PALETTE_GREY), 0);
lv_obj_align(sunrise_lbl, LV_ALIGN_LEFT_MID, 37, -10);

lv_obj_t * sunrise_time = lv_label_create(bg);
lv_label_set_text(sunrise_time, "6:43");
lv_obj_set_style_text_font(sunrise_time, &lv_font_montserrat_20, 0);
lv_obj_set_style_text_color(sunrise_time, lv_color_white(), 0);
lv_obj_align_to(sunrise_time, sunrise_lbl, LV_ALIGN_OUT_BOTTOM_MID, 0, 2);

lv_obj_t * sunset_lbl = lv_label_create(bg);
lv_label_set_text(sunset_lbl, "SUNSET");
lv_obj_set_style_text_font(sunset_lbl, &lv_font_montserrat_14, 0);
lv_obj_set_style_text_color(sunset_lbl, lv_palette_main(LV_PALETTE_GREY), 0);
lv_obj_align(sunset_lbl, LV_ALIGN_RIGHT_MID, -37, -10);

lv_obj_t * sunset_time = lv_label_create(bg);
lv_label_set_text(sunset_time, "17:37");
lv_obj_set_style_text_font(sunset_time, &lv_font_montserrat_20, 0);
lv_obj_set_style_text_color(sunset_time, lv_color_white(), 0);
lv_obj_align_to(sunset_time, sunset_lbl, LV_ALIGN_OUT_BOTTOM_MID, 0, 2);
}

#endif

```

## Uno stile di scala circolare che simula una bussola

```

#include "../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

static lv_obj_t * scale;
static lv_obj_t * label;

static const char * heading_to_cardinal(int32_t heading)
{
    /* Normalize heading to range [0, 360) */
    while(heading < 0) heading += 360;
    while(heading >= 360) heading -= 360;

    if(heading < 23) return "N";
    else if(heading < 68) return "NE";
    else if(heading < 113) return "E";
    else if(heading < 158) return "SE";
    else if(heading < 203) return "S";
    else if(heading < 248) return "SW";
    else if(heading < 293) return "W";
    else if(heading < 338) return "NW";

    return "N";
}

static void set_heading_value(void * obj, int32_t v)
{
    LV_UNUSED(obj);
    lv_scale_set_rotation(scale, 270 - v);
    lv_label_set_text_fmt(label, "%d\n%s", v, heading_to_cardinal(v));
}

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_label_dsc_t * label_dsc = lv_draw_task_get_label_dsc(draw_task);
    lv_draw_line_dsc_t * line_dsc = lv_draw_task_get_line_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        if(label_dsc) {
            if(base_dsc->id1 == 0) {
                label_dsc->color = lv_palette_main(LV_PALETTE_RED);
            }
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

        }
    }
    if(line_draw_dsc) {
        if(base_dsc->id1 == 60) {
            line_draw_dsc->color = lv_palette_main(LV_PALETTE_RED);
        }
    }
}

/**
 * A round scale style simulating a compass
 */
void lv_example_scale_12(void)
{
    scale = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale, 200, 200);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_align(scale, LV_ALIGN_CENTER);

    lv_scale_set_total_tick_count(scale, 61);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_obj_set_style_line_width(scale, 3, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 0, 360);

    static const char * custom_labels[] = {"N", "30", "60", "E", "120", "150", "S", "210", "240", "W", "300", "330", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    lv_scale_set_angle_range(scale, 360);
    lv_scale_set_rotation(scale, 270);

    lv_obj_add_event_cb(scale, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, 100);
    lv_obj_set_align(label, LV_ALIGN_CENTER);
    lv_label_set_text(label, "0%\n");
    lv_obj_set_style_text_align(label, LV_TEXT_ALIGN_CENTER, 0);

    set_heading_value(NULL, 0);

    lv_obj_t * symbol = lv_label_create(scale);
    lv_obj_set_align(symbol, LV_ALIGN_TOP_MID);
    lv_obj_set_y(symbol, 5);
    lv_label_set_text(symbol, LV_SYMBOL_UP);
    lv_obj_set_style_text_align(symbol, LV_TEXT_ALIGN_CENTER, 0);
    lv_obj_set_style_text_color(symbol, lv_palette_main(LV_PALETTE_RED), 0);

    lv_anim_t anim_scale;
    lv_anim_init(&anim_scale);
    lv_anim_set_var(&anim_scale, scale);
    lv_anim_set_exec_cb(&anim_scale, set_heading_value);
    lv_anim_set_duration(&anim_scale, 5000);
    lv_anim_set_repeat_delay(&anim_scale, 500);
    lv_anim_set_repeat_count(&anim_scale, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_reverse_duration(&anim_scale, 5000);
    lv_anim_set_reverse_delay(&anim_scale, 500);
    lv_anim_set_values(&anim_scale, 0, 360);
    lv_anim_start(&anim_scale);
}
#endif

```

### 3.7.24 Slider

#### Slider semplice

```

#include "../../lv_examples.h"
#ifndef LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);
static lv_obj_t * slider_label;

/**
 * A default slider with a label displaying the current value
 */
void lv_example_slider_1(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

    lv_obj_set_style_anim_duration(slider, 2000, 0);
    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_screen_active());
    lv_label_set_text(slider_label, "0%");

    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target_obj(e);
    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d%", (int)lv_slider_get_value(slider));
    lv_label_set_text(slider_label, buf);
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

#endif

```

## Slider con stile personalizzato

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

/***
 * Show how to style a slider.
 */
void lv_example_slider_2(void)
{
    /*Create a transition*/
    static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, 0};
    static lv_style_transition_dsc_t transition_dsc;
    lv_style_transition_dsc_init(&transition_dsc, props, lv_anim_path_linear, 300, 0, NULL);

    static lv_style_t style_main;
    static lv_style_t style_indicator;
    static lv_style_t style_knob;
    static lv_style_t style_pressed_color;
    lv_style_init(&style_main);
    lv_style_set_bg_opa(&style_main, LV_OPA_COVER);
    lv_style_set_bg_color(&style_main, lv_color_hex3(0xbbb));
    lv_style_set_radius(&style_main, LV_RADIUS_CIRCLE);
    lv_style_set_pad_ver(&style_main, -2); /*Makes the indicator larger*/

    lv_style_init(&style_indicator);
    lv_style_set_bg_opa(&style_indicator, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indicator, lv_palette_main(LV_PALETTE_CYAN));
    lv_style_set_radius(&style_indicator, LV_RADIUS_CIRCLE);
    lv_style_set_transition(&style_indicator, &transition_dsc);

    lv_style_init(&style_knob);
    lv_style_set_bg_opa(&style_knob, LV_OPA_COVER);
    lv_style_set_bg_color(&style_knob, lv_palette_main(LV_PALETTE_CYAN));
    lv_style_set_border_color(&style_knob, lv_palette_darken(LV_PALETTE_CYAN, 3));
    lv_style_set_border_width(&style_knob, 2);
    lv_style_set_radius(&style_knob, LV_RADIUS_CIRCLE);
    lv_style_set_pad_all(&style_knob, 6); /*Makes the knob larger*/
    lv_style_set_transition(&style_knob, &transition_dsc);

    lv_style_init(&style_pressed_color);
    lv_style_set_bg_color(&style_pressed_color, lv_palette_darken(LV_PALETTE_CYAN, 2));

    /*Create a slider and add the style*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_remove_style_all(slider); /*Remove the styles coming from the theme*/

    lv_obj_add_style(slider, &style_main, LV_PART_MAIN);
    lv_obj_add_style(slider, &style_indicator, LV_PART_INDICATOR);
    lv_obj_add_style(slider, &style_pressed_color, LV_PART_INDICATOR | LV_STATE_PRESSED);
    lv_obj_add_style(slider, &style_knob, LV_PART_KNOB);
    lv_obj_add_style(slider, &style_pressed_color, LV_PART_KNOB | LV_STATE_PRESSED);

    lv_obj_center(slider);
}

#endif
```

## Slider estensibile

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

#define MAX_VALUE 100
#define MIN_VALUE 0

static void slider_event_cb(lv_event_t * e);

/***
 * Show the current value when the slider is pressed by extending the drawer
 */
void lv_example_slider_3(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider;
    slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);

    lv_slider_set_mode(slider, LV_SLIDER_MODE_RANGE);
    lv_slider_set_range(slider, MIN_VALUE, MAX_VALUE);
    lv_slider_set_value(slider, 70, LV_ANIM_OFF);
    lv_slider_set_start_value(slider, 20, LV_ANIM_OFF);

    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(slider);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);

    /*Provide some extra space for the value*/
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        lv_event_set_ext_draw_size(e, 50);
    }
    else if(code == LV_EVENT_DRAW_MAIN_END) {
        if(!lv_obj_has_state(obj, LV_STATE_PRESSED)) return;

        lv_area_t slider_area;
        lv_obj_get_coords(obj, &slider_area);
        lv_area_t indic_area = slider_area;
        lv_area_set_width(&indic_area, lv_area_get_width(&slider_area) * lv_slider_get_value(obj) / MAX_VALUE);
        indic_area.x1 += lv_area_get_width(&slider_area) * lv_slider_get_left_value(obj) / MAX_VALUE;
        char buf[16];
        lv_snprintf(buf, sizeof(buf), "%d - %d", (int)lv_slider_get_left_value(obj), (int)lv_slider_get_value(obj));
        lv_text_attributes_t attributes = {0};
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

attributes.text_flags = LV_TEXT_FLAG_NONE;
attributes.max_width = LV_COORD_MAX;
attributes.letter_space = 0;
attributes.line_space = 0;

lv_point_t label_size;
lv_text_get_size(&label_size, buf, LV_FONT_DEFAULT, &attributes);
lv_area_t label_area;
label_area.x1 = 0;
label_area.x2 = label_size.x - 1;
label_area.y1 = 0;
label_area.y2 = label_size.y - 1;

lv_area_align(&indic_area, &label_area, LV_ALIGN_OUT_TOP_MID, 0, -10);

lv_draw_label_dsc_t label_draw_dsc;
lv_draw_label_dsc_init(&label_draw_dsc);
label_draw_dsc.color = lv_color_hex3(0x888);
label_draw_dsc.text = buf;
label_draw_dsc.text_local = true;
lv_layer_t * layer = lv_event_get_layer(e);
lv_draw_label(layer, &label_draw_dsc, &label_area);
}
}

#endif

```

## Slider con direzione opposta

```

#include "../../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);
static lv_obj_t * slider_label;

/***
 * Slider with opposite direction
 */
void lv_example_slider_4(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    /*Reverse the direction of the slider*/
    lv_slider_set_range(slider, 100, 0);
    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_screen_active());
    lv_label_set_text(slider_label, "0%");

    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target_obj(e);
    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d%", (int)lv_slider_get_value(slider));
    lv_label_set_text(slider_label, buf);
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

#endif

```

## 3.7.25 Span

### Span con stili personalizzati

```

#include "../../../lv_examples.h"
#if LV_USE_SPAN && LV_BUILD_EXAMPLES

static void click_event_cb(lv_event_t * e)
{
    lv_indev_t * indev = lv_event_get_indev(e);
    lv_point_t point;
    lv_indev_get_point(indev, &point);
#if LV_USE_LOG
    lv_obj_t * spans = lv_event_get_target_obj(e);
    lv_span_t * span = lv_spangroup_get_span_by_point(spans, &point);
    LV_LOG_USER("%s", span ? lv_span_get_text(span) : "NULL");
#endif
}

/***
 * Create spans and get clicked one
 */
void lv_example_span_1(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_border_width(&style, 1);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_ORANGE));
    lv_style_set_pad_all(&style, 2);

    lv_obj_t * spans = lv_spangroup_create(lv_screen_active());
    /* Setting a fixed width and height to LV_SIZE_CONTENT will make the text wrap */
    lv_obj_set_width(spans, 300);
    lv_obj_set_height(spans, LV_SIZE_CONTENT);
    lv_obj_center(spans);
    lv_obj_add_style(spans, &style, 0);
    lv_obj_add_flag(spans, LV_OBJ_FLAG_CLICKABLE);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_spangroup_set_align(spans, LV_TEXT_ALIGN_LEFT);
lv_spangroup_set_overflow(spans, LV_SPAN_OVERFLOW_CLIP);
lv_spangroup_set_indent(spans, 20);

lv_span_t * span = lv_spangroup_add_span(spans);
lv_span_set_text(span, "China is a beautiful country.");
lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_RED));
lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_UNDERLINE);
lv_style_set_text_opa(lv_span_get_style(span), LV_OPA_50);

span = lv_spangroup_add_span(spans);
lv_span_set_text_static(span, "good good study, day day up.");
#if LV_FONT_MONTSERRAT_24
lv_style_set_text_font(lv_span_get_style(span), &lv_font_montserrat_24);
#endif
lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_GREEN));

span = lv_spangroup_add_span(spans);
lv_span_set_text_static(span, "LVGL is an open-source graphics library.");
lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_BLUE));

span = lv_spangroup_add_span(spans);
lv_span_set_text_static(span, "the boy no name.");
lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_GREEN));
#if LV_FONT_MONTSERRAT_20
lv_style_set_text_font(lv_span_get_style(span), &lv_font_montserrat_20);
#endif
lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_UNDERLINE);

span = lv_spangroup_add_span(spans);
lv_span_set_text(span, "I have a dream that hope to come true.");
lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_STRIKETHROUGH);

lv_spangroup_refresh(spans);

lv_obj_add_event_cb(spans, click_event_cb, LV_EVENT_CLICKED, NULL);
}

#endif

```

## 3.7.26 Spinbox

### Spinbox semplice

```

#include "../lv_examples.h"
#if LV_USE_SPINBOX && LV_BUILD_EXAMPLES

static lv_obj_t * spinbox;
static void lv_spinbox_increment_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_increment(spinbox);
    }
}
static void lv_spinbox_decrement_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_decrement(spinbox);
    }
}
void lv_example_spinbox_1(void)
{
    spinbox = lv_spinbox_create(lv_screen_active());
    lv_spinbox_set_range(spinbox, -1000, 25000);
    lv_spinbox_set_digit_format(spinbox, 5, 2);
    lv_spinbox_step_prev(spinbox);
    lv_obj_set_width(spinbox, 100);
    lv_obj_center(spinbox);

    int32_t h = lv_obj_get_height(spinbox);

    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, h, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_RIGHT_MID, 5, 0);
    lv_obj_set_style_bg_image_src(btn, LV_SYMBOL_PLUS, 0);
    lv_obj_add_event_cb(btn, lv_spinbox_increment_event_cb, LV_EVENT_ALL, NULL);

    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, h, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_LEFT_MID, -5, 0);
    lv_obj_set_style_bg_image_src(btn, LV_SYMBOL_MINUS, 0);
    lv_obj_add_event_cb(btn, lv_spinbox_decrement_event_cb, LV_EVENT_ALL, NULL);
}

#endif

```

## 3.7.27 Spinner

### Spinner semplice

```

#include "../lv_examples.h"
#if LV_USE_SPINNER && LV_BUILD_EXAMPLES

void lv_example_spinner_1(void)
{
    /*Create a spinner*/

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * spinner = lv_spinner_create(lv_screen_active());
lv_obj_set_size(spinner, 100, 100);
lv_obj_center(spinner);
lv_spinner_set_anim_params(spinner, 10000, 200);
}

#endif

```

## 3.7.28 Switch

### Switch semplice

```

#include "../../../lv_examples.h"
#if LV_USE_SWITCH && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_UNUSED(obj);
        LV_LOG_USER("State: %s\n", lv_obj_has_state(obj, LV_STATE_CHECKED) ? "On" : "Off");
    }
}

void lv_example_switch_1(void)
{
    lv_obj_set.flex_flow(lv_screen_active(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set.flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * sw;

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_add_flag(sw, LV_OBJ_FLAG_EVENT_BUBBLE);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_state(sw, LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
}

#endif

```

### Orientamento dello Switch

```

#include "../../../lv_examples.h"
#if LV_USE_SWITCH && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_UNUSED(obj);
        LV_LOG_USER("State: %s\n", lv_obj_has_state(obj, LV_STATE_CHECKED) ? "On" : "Off");
    }
}

void lv_example_switch_2(void)
{
    lv_obj_set.flex_flow(lv_screen_active(), LV_FLEX_FLOW_ROW);
    lv_obj_set.flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * sw;

    sw = lv_switch_create(lv_screen_active());
    lv_obj_set.size(sw, 30, 60);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_set.size(sw, 30, 60);
    lv_switch_set_orientation(sw, LV_SWITCH_ORIENTATION_VERTICAL);
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
}

#endif

```

## 3.7.29 Tabella

### Tabella semplice

```

#include "../../../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
/*If the cells are drawn...*/
if(base_dsc->part == LV_PART_ITEMS) {
    uint32_t row = base_dsc->id1;
    uint32_t col = base_dsc->id2;

    /*Make the texts in the first cell center aligned*/
    if(row == 0) {
        lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
        if(label_draw_dsc) {
            label_draw_dsc->align = LV_TEXT_ALIGN_CENTER;
        }
        lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
        if(fill_draw_dsc) {
            fill_draw_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_BLUE), fill_draw_dsc->color, LV_OPA_20);
            fill_draw_dsc->opa = LV_OPA_COVER;
        }
    }
    /*In the first column align the texts to the right*/
    else if(col == 0) {
        lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
        if(label_draw_dsc) {
            label_draw_dsc->align = LV_TEXT_ALIGN_RIGHT;
        }
    }
}

/*Make every 2nd row grayish*/
if((row != 0 && row % 2) == 0) {
    lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
    if(fill_draw_dsc) {
        fill_draw_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_GREY), fill_draw_dsc->color, LV_OPA_10);
        fill_draw_dsc->opa = LV_OPA_COVER;
    }
}
}

void lv_example_table_1(void)
{
    lv_obj_t * table = lv_table_create(lv_screen_active());

    /*Fill the first column*/
    lv_table_set_cell_value(table, 0, 0, "Name");
    lv_table_set_cell_value(table, 1, 0, "Apple");
    lv_table_set_cell_value(table, 2, 0, "Banana");
    lv_table_set_cell_value(table, 3, 0, "Lemon");
    lv_table_set_cell_value(table, 4, 0, "Grape");
    lv_table_set_cell_value(table, 5, 0, "Melon");
    lv_table_set_cell_value(table, 6, 0, "Peach");
    lv_table_set_cell_value(table, 7, 0, "Nuts");

    /*Fill the second column*/
    lv_table_set_cell_value(table, 0, 1, "Price");
    lv_table_set_cell_value(table, 1, 1, "$7");
    lv_table_set_cell_value(table, 2, 1, "$4");
    lv_table_set_cell_value(table, 3, 1, "$6");
    lv_table_set_cell_value(table, 4, 1, "$2");
    lv_table_set_cell_value(table, 5, 1, "$5");
    lv_table_set_cell_value(table, 6, 1, "$1");
    lv_table_set_cell_value(table, 7, 1, "$9");

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_height(table, 200);
    lv_obj_center(table);

    /*Add an event callback to to apply some custom drawing*/
    lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(table, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
}

#endif

```

## Elenco leggero da tabella

```

#include "../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

#define ITEM_CNT 200

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    /*If the cells are drawn...*/
    if(base_dsc->part == LV_PART_ITEMS && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_FILL) {
        /*Draw the background*/
        bool chk = lv_table_has_cell_ctrl(obj, base_dsc->id1, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
        lv_draw_rect_dsc_t rect_dsc;
        lv_draw_rect_dsc_init(&rect_dsc);
        rect_dsc.bg_color = chk ? lv_theme_get_color_primary(obj) : lv_palette_lighten(LV_PALETTE_GREY, 2);
        rect_dsc.radius = LV_RADIUS_CIRCLE;

        lv_area_t sw_area;
        sw_area.x1 = 0;
        sw_area.x2 = 40;
        sw_area.y1 = 0;
        sw_area.y2 = 24;
        lv_area_t draw_task_area;
        lv_draw_task_get_area(draw_task, &draw_task_area);
        lv_area_align(&draw_task_area, &sw_area, LV_ALIGN_RIGHT_MID, -15, 0);
        lv_draw_rect(base_dsc->layer, &rect_dsc, &sw_area);

        /*Draw the knob*/
        rect_dsc.bg_color = lv_color_white();
        lv_area_t knob_area;
        knob_area.x1 = 0;

```

(continues on next page)

(continua dalla pagina precedente)

```

knob_area.x2 = 18;
knob_area.y1 = 0;
knob_area.y2 = 18;
if(chk) {
    lv_area_align(&sw_area, &knob_area, LV_ALIGN_RIGHT_MID, -3, 0);
}
else {
    lv_area_align(&sw_area, &knob_area, LV_ALIGN_LEFT_MID, 3, 0);
}
lv_draw_rect(base_dsc->layer, &rect_dsc, &knob_area);
}

static void change_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    uint32_t col;
    uint32_t row;
    lv_table_get_selected_cell(obj, &row, &col);
    bool chk = lv_table_has_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
    if(chk) lv_table_clear_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
    else lv_table_set_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
}

/**
 * A very light-weighted list created from table
 */
void lv_example_table_2(void)
{
    /*Measure memory usage*/
    lv_mem_monitor_t mon1;
    lv_mem_monitor(&mon1);

    uint32_t t = lv_tick_get();

    lv_obj_t * table = lv_table_create(lv_screen_active());

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_size(table, LV_SIZE_CONTENT, 200);

    lv_table_set_column_width(table, 0, 150);
    lv_table_set_row_count(table, ITEM_CNT); /*Not required but avoids a lot of memory reallocation lv_table_set_set_value*/
    lv_table_set_column_count(table, 1);

    /*Don't make the cell pressed, we will draw something different in the event*/
    lv_obj_remove_style(table, NULL, LV_PART_ITEMS | LV_STATE_PRESSED);

    uint32_t i;
    for(i = 0; i < ITEM_CNT; i++) {
        lv_table_set_cell_value_fmt(table, i, 0, "Item %" LV_PRIu32, i + 1);
    }

    lv_obj_align(table, LV_ALIGN_CENTER, 0, -20);

    /*Add an event callback to to apply some custom drawing*/
    lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_event_cb(table, change_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_add_flag(table, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    lv_mem_monitor_t mon2;
    lv_mem_monitor(&mon2);

    size_t mem_used = mon1.free_size - mon2.free_size;
    uint32_t elaps = lv_tick_elaps(t);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text_fmt(label, "%" LV_PRIu32 " items were created in %" LV_PRIu32 " ms\n"
                           "using %zu bytes of memory",
                           (uint32_t)ITEM_CNT, elaps, mem_used);

    lv_obj_align(label, LV_ALIGN_BOTTOM_MID, 0, -10);
}
#endif

```

### 3.7.30 Tabview

#### Tabview Semplice

```

#include "../lv_examples.h"
#if LV_USE_TABVIEW && LV_BUILD_EXAMPLES

void lv_example_tabview_1(void)
{
    /*Create a Tab view object*/
    lv_obj_t * tabview;
    tabview = lv_tabview_create(lv_screen_active());

    /*Add 3 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t * tab1 = lv_tabview_add_tab(tabview, "Tab 1");
    lv_obj_t * tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t * tab3 = lv_tabview_add_tab(tabview, "Tab 3");

    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "This the first tab\n\n"
                           "If the content\n"
                           "of a tab\n"
                           "becomes too\n"
                           "longer\n"
                           "than the\n"
                           "container\n"
                           "then it\n"
                           "automatically\n"
                           "becomes\n");
}

```

(continues on next page)

(continua dalla pagina precedente)

```

"scrollable.\n"
"\n"
"\n"
"\n"
"Can you see it?");

label = lv_label_create(tab2);
lv_label_set_text(label, "Second tab");

label = lv_label_create(tab3);
lv_label_set_text(label, "Third tab");

lv_obj_scroll_to_view_recursive(label, LV_ANIM_ON);

}
#endif

```

### Tab a sinistra, con stile e nessuno scorrimento

```

#include "../../../lv_examples.h"
#if LV_USE_TABVIEW && LV_BUILD_EXAMPLES

/*A vertical tab view with disabled scrolling and some styling*/
void lv_example_tabview_2(void)
{
    /*Create a Tab view object*/
    lv_obj_t * tabview;
    uint32_t tab_count = 0;
    uint32_t i = 0;

    tabview = lv_tabview_create(lv_screen_active());
    lv_tabview_set_tab_bar_position(tabview, LV_DIR_LEFT);
    lv_tabview_set_tab_bar_size(tabview, 80);

    lv_obj_set_style_bg_color(tabview, lv_palette_lighten(LV_PALETTE_RED, 2), 0);

    lv_obj_t * tab_buttons = lv_tabview_get_tab_bar(tabview);
    lv_obj_set_style_bg_color(tab_buttons, lv_palette_darken(LV_PALETTE_GREY, 3), 0);
    lv_obj_set_style_text_color(tab_buttons, lv_palette_lighten(LV_PALETTE_GREY, 5), 0);

    /*Add 5 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t * tab1 = lv_tabview_add_tab(tabview, "Tab 1");
    lv_obj_t * tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t * tab3 = lv_tabview_add_tab(tabview, "Tab 3");
    lv_obj_t * tab4 = lv_tabview_add_tab(tabview, "Tab 4");
    lv_obj_t * tab5 = lv_tabview_add_tab(tabview, "Tab 5");

    tab_count = lv_tabview_get_tab_count(tabview);
    for(i = 0; i < tab_count; i++) {
        lv_obj_t * button = lv_obj_get_child(tab_buttons, i);
        lv_obj_set_style_border_side(button, LV_BORDER_SIDE_RIGHT, LV_PART_MAIN | LV_STATE_CHECKED);
    }
    lv_obj_set_style_bg_color(tab2, lv_palette_lighten(LV_PALETTE_AMBER, 3), 0);
    lv_obj_set_style_bg_opa(tab2, LV_OPA_COVER, 0);

    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "First tab");

    label = lv_label_create(tab2);
    lv_label_set_text(label, "Second tab");

    label = lv_label_create(tab3);
    lv_label_set_text(label, "Third tab");

    label = lv_label_create(tab4);
    lv_label_set_text(label, "Fourth tab");

    label = lv_label_create(tab5);
    lv_label_set_text(label, "Fifth tab");

    lv_obj_remove_flag(lv_tabview_get_content(tabview), LV_OBJ_FLAG_SCROLLABLE);
}
#endif

```

### 3.7.31 Area di testo

#### Area di testo semplice

```

#include "../../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_BUILD_EXAMPLES

static void textarea_event_handler(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target_obj(e);
    LV_UNUSED(ta);
    LV_LOG_USER("Enter was pressed. The current text is: %s", lv_textarea_get_text(ta));
}

static void btm_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * ta = (lv_obj_t *)lv_event_get_user_data(e);
    const char * txt = lv_buttonmatrix_get_button_text(obj, lv_buttonmatrix_get_selected_button(obj));

    if(lv_strcmp(txt, LV_SYMBOL_BACKSPACE) == 0) lv_textarea_delete_char(ta);
    else if(lv_strcmp(txt, LV_SYMBOL_NEW_LINE) == 0) lv_obj_send_event(ta, LV_EVENT_READY, NULL);
    else lv_textarea_add_text(ta, txt);
}

void lv_example_textarea_1(void)

```

(continues on next page)

(continua dalla pagina precedente)

```
{
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_one_line(ta, true);
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, 10);
    lv_obj_add_event_cb(ta, textarea_event_handler, LV_EVENT_READY, ta);
    lv_obj_add_state(ta, LV_STATE_FOCUSED); /*To be sure the cursor is visible*/

    static const char * btnm_map[] = {"1", "2", "3", "\n",
        "4", "5", "6", "\n",
        "7", "8", "9", "\n",
        LV_SYMBOL_BACKSPACE, "0", LV_SYMBOL_NEW_LINE, ""
    };

    lv_obj_t * btnm = lv_buttonmatrix_create(lv_screen_active());
    lv_obj_set_size(btnm, 200, 150);
    lv_obj_align(btnm, LV_ALIGN_BOTTOM_MID, 0, -10);
    lv_obj_add_event_cb(btnm, btnm_event_handler, LV_EVENT_VALUE_CHANGED, ta);
    lv_obj_remove_flag(btnm, LV_OBJ_FLAG_CLICK_FOCUSABLE); /*To keep the text area focused on button clicks*/
    lv_buttonmatrix_set_map(btnm, btnm_map);
}

#endif
```

## Area di testo con campo password

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;

void lv_example_textarea_2(void)
{
    /*Create the password box*/
    lv_obj_t * pwd_ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_text(pwd_ta, "");
    lv_textarea_set_password_mode(pwd_ta, true);
    lv_textarea_set_one_line(pwd_ta, true);
    lv_obj_set_width(pwd_ta, lv_pct(40));
    lv_obj_set_pos(pwd_ta, 5, 20);
    lv_obj_add_event_cb(pwd_ta, ta_event_cb, LV_EVENT_ALL, NULL);

    /*Create a label and position it above the text box*/
    lv_obj_t * pwd_label = lv_label_create(lv_screen_active());
    lv_label_set_text(pwd_label, "Password:");
    lv_obj_align_to(pwd_label, pwd_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create the one-line mode text area*/
    lv_obj_t * text_ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_one_line(text_ta, true);
    lv_textarea_set_password_mode(text_ta, false);
    lv_obj_set_width(text_ta, lv_pct(40));
    lv_obj_add_event_cb(text_ta, ta_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_align(text_ta, LV_ALIGN_TOP_RIGHT, -5, 20);

    /*Create a label and position it above the text box*/
    lv_obj_t * oneline_label = lv_label_create(lv_screen_active());
    lv_label_set_text(oneline_label, "Text:");
    lv_obj_align_to(oneline_label, text_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create a keyboard*/
    kb = lv_keyboard_create(lv_screen_active());
    lv_obj_set_size(kb, LV_HOR_RES, LV_VER_RES / 2);

    lv_keyboard_set_textarea(kb, pwd_ta); /*Focus it on one of the text areas to start*/

    /*The keyboard will show Arabic characters if they are enabled */
#if LV_USE_ARABIC_PERSIAN_CHARS && LV_FONT_DEJAVU_16_PERSIAN_HEBREW
    lv_obj_set_style_text_font(kb, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_style_text_font(text_ta, &lv_font_dejavu_16_persian_hebrew, 0);
    lv_obj_set_style_text_font(pwd_ta, &lv_font_dejavu_16_persian_hebrew, 0);
#endif
}

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED || code == LV_EVENT_FOCUSED) {
        /*Focus on the clicked text area*/
        if(kb != NULL) lv_keyboard_set_textarea(kb, ta);
    }

    else if(code == LV_EVENT_READY) {
        LV_LOG_USER("Ready, current text: %s", lv_textarea_get_text(ta));
    }
}

#endif
```

## Formattazione automatica del testo

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;

/***
 * Automatically format text like a clock. E.g. "12:34"
 * Add the ':' automatically.
*/
```

(continues on next page)

(continua dalla pagina precedente)

```
/*
void lv_example_textarea_3(void)
{
    /*Create the text area*/
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_obj_add_event_cb(ta, ta_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_textarea_set_accepted_chars(ta, "0123456789:");
    lv_textarea_set_max_length(ta, 5);
    lv_textarea_set_one_line(ta, true);
    lv_textarea_set_text(ta, "");

    /*Create a keyboard*/
    kb = lv_keyboard_create(lv_screen_active());
    lv_obj_set_size(kb, LV_HOR_RES, LV_VER_RES / 2);
    lv_keyboard_set_mode(kb, LV_KEYBOARD_MODE_NUMBER);
    lv_keyboard_set_textarea(kb, ta);
}

static void ta_event_cb(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target_obj(e);
    const char * txt = lv_textarea_get_text(ta);
    if(txt[0] >= '0' && txt[0] <= '9' &&
       txt[1] >= '0' && txt[1] <= '9' &&
       txt[2] != ':') {
        lv_textarea_set_cursor_pos(ta, 2);
        lv_textarea_add_char(ta, ':');
    }
}
#endif
```

## Stile del cursore dell'area di testo

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void create_styled_textarea_cursor(const char * txt, lv_coord_t y_ofs, lv_style_t * cursor_style)
{
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_text(ta, txt);
    lv_obj_set_width(ta, 280);
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, y_ofs);
    lv_textarea_set_one_line(ta, true);
    lv_obj_add_state(ta, LV_STATE_FOCUSED);
    lv_obj_add_style(ta, cursor_style, LV_PART_CURSOR | LV_STATE_FOCUSED);
    lv_textarea_set_cursor_pos(ta, 0);
}

void lv_example_textarea_4(void)
{
    static lv_style_t style_simple, style_block, style_underline;

    /* Thin left bar cursor (simple) */
    lv_style_init(&style_simple);
    lv_style_set_border_color(&style_simple, lv_palette_main(LV_PALETTE_RED));

    /* Underline cursor */
    lv_style_init(&style_underline);
    lv_style_set_bg_opa(&style_underline, LV_OPA_TRANSP);
    lv_style_set_border_color(&style_underline, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_side(&style_underline, LV_BORDER_SIDE_BOTTOM);
    lv_style_set_pad_hor(&style_underline, 1); /* set width of cursor using pad */
    lv_style_set_border_width(&style_underline, 3); /* set thickness of underline cursor */

    /* Full block cursor with many styles */
    lv_style_init(&style_block);
    lv_style_set_bg_opa(&style_block, LV_OPA_COVER);
    lv_style_set_bg_color(&style_block, lv_palette_main(LV_PALETTE_ORANGE));
    lv_style_set_bg_grad_color(&style_block, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_bg_grad_dir(&style_block, LV_GRAD_DIR_VER);
    lv_style_set_border_color(&style_block, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_border_side(&style_block, LV_BORDER_SIDE_FULL);
    lv_style_set_border_width(&style_block, 1);
    lv_style_set_radius(&style_block, 4);
    lv_style_set_text_color(&style_block, lv_color_white());
    lv_style_set_pad_all(&style_block, 1); /* set width of cursor using pad */

    /* Create 3 independent textareas, each with a unique styled cursor */
    create_styled_textarea_cursor("This is a simple red cursor", 10, &style_simple);
    create_styled_textarea_cursor("This is an underline blue cursor", 110, &style_underline);
    create_styled_textarea_cursor("This is a complex block cursor", 60, &style_block);
}

#endif
```

## 3.7.32 Tileview

### Tileview con contenuto

```
#include "../../lv_examples.h"
#if LV_USE_TILEVIEW && LV_BUILD_EXAMPLES

/***
 * Create a 2x2 tile view and allow scrolling only in an "L" shape.
 * Demonstrate scroll chaining with a long list that
 * scrolls the tile view when it can't be scrolled further.
 */
void lv_example_tileview_1(void)
{
    lv_obj_t * tv = lv_tileview_create(lv_screen_active());
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Tile1: just a label*/
lv_obj_t * tile1 = lv_tileview_add_tile(tv, 0, 0, LV_DIR_BOTTOM);
lv_obj_t * label = lv_label_create(tile1);
lv_label_set_text(label, "Scroll down");
lv_obj_center(label);

/*Tile2: a button*/
lv_obj_t * tile2 = lv_tileview_add_tile(tv, 0, 1, (lv_dir_t)(LV_DIR_TOP | LV_DIR_RIGHT));
lv_obj_t * btn = lv_button_create(tile2);

label = lv_label_create(btn);
lv_label_set_text(label, "Scroll up or right");

lv_obj_set_size(btn, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_center(btn);

/*Tile3: a list*/
lv_obj_t * tile3 = lv_tileview_add_tile(tv, 1, 1, LV_DIR_LEFT);
lv_obj_t * list = lv_list_create(tile3);
lv_obj_set_size(list, LV_PCT(100), LV_PCT(100));

lv_list_add_button(list, NULL, "One");
lv_list_add_button(list, NULL, "Two");
lv_list_add_button(list, NULL, "Three");
lv_list_add_button(list, NULL, "Four");
lv_list_add_button(list, NULL, "Five");
lv_list_add_button(list, NULL, "Six");
lv_list_add_button(list, NULL, "Seven");
lv_list_add_button(list, NULL, "Eight");
lv_list_add_button(list, NULL, "Nine");
lv_list_add_button(list, NULL, "Ten");

}

#endif

```

### 3.7.33 Window

#### Window semplice

```

#include "../../lv_examples.h"
#if LV_USE_WIN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    LV_UNUSED(obj);
    LV_LOG_USER("Button %d clicked", (int)lv_obj_get_index(obj));
}

void lv_example_win_1(void)
{
    lv_obj_t * win = lv_win_create(lv_screen_active());
    lv_obj_t * btn;
    btn = lv_win_add_button(win, LV_SYMBOL_LEFT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_win_add_title(win, "A title");

    btn = lv_win_add_button(win, LV_SYMBOL_RIGHT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    btn = lv_win_add_button(win, LV_SYMBOL_CLOSE, 60);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_obj_t * cont = lv_win_get_content(win); /*Content can be added here*/
    lv_obj_t * label = lv_label_create(cont);
    lv_label_set_text(label, "This is\n"
                           "a pretty\n"
                           "long text\n"
                           "to see how\n"
                           "the window\n"
                           "becomes\n"
                           "scrollable.\n"
                           "\n"
                           "\n"
                           "Some more\n"
                           "text to be\n"
                           "sure it\n"
                           "overflows. :)");
}
#endif

```

# CAPITOLO 4

## Integrazione e Driver

### 4.1 Aggiungere LVGL al Proprio Progetto

#### 4.1.1 Ottenerne LVGL

LVGL è disponibile su GitHub: <https://github.com/lvgl/lvgl>.

Lo si può clonare o eseguire il [Download](#) dell'ultima versione della libreria da GitHub.

La libreria grafica vera e propria si trova nella directory `lvgl`. Contiene diverse directory, ma per usare LVGL sono necessari solo i file `.c` e `.h` nella directory `src`, più `lvgl/lvgl.h`, e `lvgl/lv_version.h`.

#### Demo ed Esempi

La directory `lvgl` contiene anche le directory `examples` e `demos`. Se il progetto necessita di esempi e/o demo, aggiungere queste directory al progetto. Se `make` o `CMake` gestiscono le directory `examples` e `demos`, non è richiesta alcuna azione aggiuntiva.

#### 4.1.2 Compilazione di LVGL

##### Make e CMake

LVGL supporta anche i sistemi di compilazione `make` e `CMake` pronti all'uso. Per aggiungere LVGL al sistema di compilazione basato su Makefile, si aggiungono queste righe al Makefile principale:

```
LVGL_DIR_NAME ?= lvgl      #The name of the lvgl folder (change this if you have renamed it)
LVGL_DIR ?= ${shell pwd}    #The path where the lvgl folder is
include $(LVGL_DIR)/$(LVGL_DIR_NAME)/lvgl.mk
```

Per l'integrazione con CMake, consultare questa sezione della [Documentazione](#).

##### Build gestite

TODO [da fare]

#### 4.1.3 Configurazione

##### lv\_conf.h

## Creazione di lv\_conf.h

Quando si imposta il progetto per la prima volta, si deve copiare `lvgl/lv_conf_template.h` in `lv_conf.h` accanto alla cartella `lvgl`. Si modifica il primo `#if 0` to 1 per abilitare il contenuto del file e si imposta la definizione `LV_COLOR_DEPTH` in modo che si allinei alla profondità di colore utilizzata dal tuo pannello di visualizzazione. Per i dettagli vedere i commenti in `lv_conf.h`.

Il layout dei file dovrebbe apparire così:

```
lvgl/
lv_conf.h
other files and folders in your project
```

In alternativa, `lv_conf.h` può essere copiato in un'altra posizione, ma in tal caso è necessario aggiungere la definizione `LV_CONF_INCLUDE_SIMPLE` alle opzioni del compilatore (ad esempio `-DLV_CONF_INCLUDE_SIMPLE` per il compilatore GCC) e impostare manualmente il path dell'include (ad esempio `-I../../include/gui`). In questo caso, LVGL tenterà di includere `lv_conf.h` semplicemente con `#include "lv_conf.h"`.

È anche possibile utilizzare un nome diverso per `lv_conf.h`. Il path personalizzato può essere impostato tramite la definizione `LV_CONF_PATH`. Per esempio `-DLV_CONF_PATH="/home/joe/my_project/my_custom_conf.h"`. Se questa definizione è settata, `LV_CONF_SKIP` si presume sia 0. Si noti che, quando si definisce `LV_CONF_PATH`, è necessario assicurarsi che sia definito come una stringa, altrimenti verrà generato un errore di compilazione.

Se `LV_CONF_SKIP` è definito, LVGL non tenterà di includere `lv_conf.h`. Invece, è possibile passare le definizioni di configurazione utilizzando le opzioni di compilazione. Ad esempio `"-DLV_COLOR_DEPTH=32 -DLV_USE_BUTTON=1"`. Le opzioni non impostate otterranno un valore predefinito che è lo stesso del contenuto di `lv_conf_template.h`.

## Impostazioni di Configurazione

Una volta installato il file `lv_conf.h`, è possibile modificare questo header per configurare il comportamento di LVGL, disabilitare moduli e funzionalità inutilizzati, regolare la dimensione dei buffer, ecc.

I commenti in `lv_conf.h` spiegano il significato di ciascuna impostazione. Si deve impostare almeno `LV_COLOR_DEPTH` in base alla profondità di colore del display. Nota: gli esempi e le demo devono essere esplicitamente abilitati in `lv_conf.h` se sono necessarie.

TODO: Aggiungere tutto ciò che riguarda il file `lv_conf.h` e il suo contenuto.

## Istanze Multiple di LVGL

È possibile eseguire più istanze indipendenti di LVGL. Per abilitare la funzionalità multi-istanza, impostare `LV_GLOBAL_CUSTOM` in `lv_conf.h` e fornire una funzione personalizzata a `lv_global_default()` utilizzando `_thread` o `pthread_key_t`. Questo consentirà l'esecuzione di più istanze di LVGL memorizzando le variabili globali di LVGL in TLS (Thread-Local Storage).

Ad esempio:

```
lv_global_t * lv_global_default(void)
{
    static __thread lv_global_t lv_global;
    return &lv_global;
}
```

## Kconfig

LVGL può essere configurato anche tramite Kconfig. Per ora, questa funzionalità è disponibile solo tramite cmake. Internamente, utilizza la porta Python di Kconfig `kconfiglib` per poterlo utilizzare su diverse piattaforme. `kconfiglib` offre l'API Python e alcuni comandi CLI. Ecco un elenco di alcuni comandi utili:

- `menuconfig`: Apre un'interfaccia a menù della console per modificare i valori di configurazione.
- `guiconfig` (richiede `tkinter`): Apre un'interfaccia grafica per modificare i valori di configurazione.
- `savedefconfig`: Salva il file `.config` corrente come `defconfig`, elencando solo i valori non di default.

- **alldefconfig**: Crea un file .config con tutti i valori di default.
- **genconfig**: Genera un header C dalla configurazione, seguendo il formato `autoconf.h`.

## Prerequisiti

Installare i prerequisiti usando `scripts/install_prerequisites.sh/bat`.

### Creazione della configurazione (.config)

A questo punto, il comando `menuconfig` dovrebbe essere disponibile:

```
cd <lvgl_repo>
menuconfig
```

Apportare le modifiche alla configurazione e uscire premendo `Esc` o `Q`, quindi salvare la configurazione. Il file `.config` è stato creato e contiene l'elenco dei valori di configurazione.

### Configurazione con cmake

Una volta creato il file ``.config``, eseguire cmake con il flag `-DLV_USE_KCONFIG=ON`:

```
cd <lvgl_repo>
cmake -B build -DLV_USE_KCONFIG=ON
cmake --build build
```

Per utilizzare un file `defconfig`, è possibile utilizzare il flag `-DLV_DEFCONFIG_PATH=<path_to_defconfig>`:

```
cd <lvgl_repo>
cmake -B build -DLV_USE_KCONFIG=ON -DLV_DEFCONFIG_PATH=<path_to_defconfig>
cmake --build build
```

Alcuni file `defconfig` sono disponibili nella cartella `configs/defconfigs`.

### Salvataggio di un defconfig

È possibile salvare un `defconfig` utilizzando il comando `savedefconfig`:

```
cd <lvgl_repo>
menuconfig # make your changes to the default config
savedefconfig
cp defconfig configs/defconfigs/my_custom_defconfig # save it where you want
# Then use it to build LVGL
cmake -B build -DLV_USE_KCONFIG=ON -DLV_DEFCONFIG_PATH=configs/defconfigs/my_custom_defconfig
cmake --build build
```

## 4.1.4 Collegamento di LVGL all'Hardware

### Inizializzazione di LVGL

Dopo aver:

- *acquisito LVGL*,
- aggiunto i file LVGL appropriati al progetto e
- *creato un file lv\_conf.h* per il progetto,

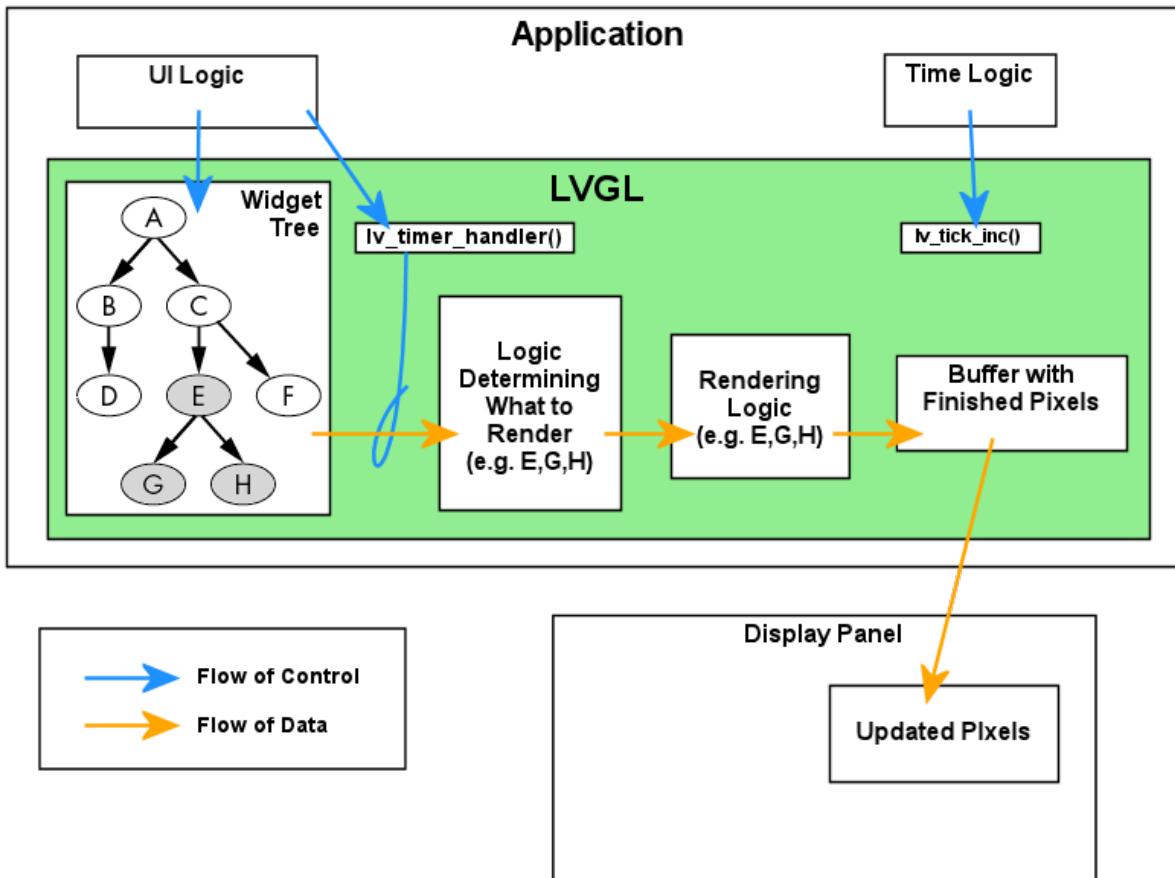
saranno necessari alcuni passaggi aggiuntivi per avviare il progetto con LVGL.

1. Inizializzare LVGL una volta all'inizio dell'esecuzione del sistema chiamando `lv_init()`. Questa operazione deve essere eseguita prima di qualsiasi altra chiamata a LVGL.
2. Inizializzare i driver.
3. Collegare *Tick Interface*.
4. Collegare *Interfaccia al Display*.
5. Collegare *Interfaccia al Device di Input*.

6. Gestire le attività LVGL relative al tempo chiamando `lv_timer_handler()` ogni pochi millisecondi per gestire i timer LVGL. Consultare [Gestione del Timer](#) per diversi modi per farlo.
7. Facoltativamente, si imposta un tema con `lv_display_set_theme()`.
8. Dopodiché #include "lvgl/lvgl.h" nei file sorgente ovunque ci sia bisogno di utilizzare le funzioni LVGL.

### Tick Interface

LVGL ha bisogno di conoscere l'ora esatta (ovvero il tempo trascorso in millisecondi) per tutte le attività in cui il tempo è un fattore determinante: aggiornamento dei display, lettura dell'input dell'utente, attivazione di eventi, animazioni, ecc.



Esistono due modi per fornire queste informazioni a LVGL:

1. Fornire a LVGL una funzione di callback per recuperare i millisecondi di sistema trascorsi chiamando `lv_tick_set_cb(my_get_milliseconds)`. `my_get_milliseconds()` deve restituire il numero di millisecondi trascorsi dall'avvio del sistema. Molte piattaforme dispongono di funzioni native utilizzabili così come sono. Ad esempio:

- SDL: `lv_tick_set_cb(SDL_GetTicks);`
- Arduino: `lv_tick_set_cb(my_tick_get_cb);`, dove `my_tick_get_cb` è: `static uint32_t my_tick_get_cb(void) { return millis(); }`
- FreeRTOS: `lv_tick_set_cb(xTaskGetTickCount);`
- STM32: `lv_tick_set_cb(HAL_GetTick);`
- ESP32: `lv_tick_set_cb(my_tick_get_cb);`, dove `my_tick_get_cb` è un wrapper per `esp_timer_get_time() / 1000;`

2. Chiamare periodicamente `lv_tick_inc(x)`, dove x rappresenta i millisecondi trascorsi dall'ultima chiamata. Se `lv_tick_inc()` viene chiamato da un ISR, dovrebbe provenire da un interrupt ad alta priorità o da un interrupt che non può essere perso quando il sistema è sotto carico elevato.

### Nota

`lv_tick_inc()` è solo una delle due funzioni LVGL che possono essere chiamate da un interrupt se la scrittura su un valore `uint32_t` è atomica sulla piattaforma. Vedere di seguito la sezione *Considerazioni sui Thread* per saperne di più.

In entrambi i casi, la scrittura del valore Tick `uint32_t` dev'essere *atomic*, il che è solitamente il caso di una piattaforma a 32 o 64 bit. Se si utilizza un sistema a 16 bit (per cui l'aggiornamento del valore Tick non è atomico) e la piattaforma utilizza il set di istruzioni Harvard, è possibile impostare una funzione come questa come callback passata a `lv_tick_set_cb(my_get_milliseconds)`:

```
/**  
 * @brief Safe read from 'elapsed_power_on_time_in_ms'  
 */  
uint32_t my_get_milliseconds()  
{  
    register uint32_t u32result;  
    /* Disable priority 1-6 interrupts for 2 Fcys. */  
    __builtin_disi(2);  
    u32result = elapsed_power_on_time_in_ms; /* Cost: 2 Fcys */  
    /* Generally looks like this in assembly:  
     *   mov    elapsed_power_on_time_in_ms, W0  
     *   mov    0x7898, W1  
     * requiring exactly 2 clock cycles.  
     * Now value is copied to register pair W0:W1  
     * where it can be written to any destination. */  
    return u32result;  
}
```

## Affidabilità

L'avanzamento del valore di tick dovrebbe essere eseguito in modo tale che la sua temporizzazione sia affidabile e non dipenda da nulla che consumi una quantità di tempo sconosciuta. Un esempio di cosa *non* fare: questo può "sembrare" funzionare, ma la temporizzazione di LVGL sarà errata perché il tempo di esecuzione di `lv_timer_handler()` varia da chiamata a chiamata e quindi il ritardo che introduce non può essere noto.

```
// Bad idea  
lv_timer_handler();  
lv_tick_inc(5);  
my_delay_ms(5);
```

## Interfaccia al Display

LVGL deve essere dotato di informazioni su ciascun pannello di visualizzazione che si desidera utilizzare. In particolare:

- il formato e la dimensione dei pixel (*Creating a Display*),
- dove renderizzare i pixel (*Draw Buffer(s)*), e
- come inviargli tali pixel renderizzati (*Flush Callback*).

Consultare i rispettivi link per informazioni su come fornire queste informazioni a LVGL.

## Interfaccia al Device di Input

LVGL deve sapere come ricevere input da tutti i dispositivi di input utente che verranno utilizzati nel progetto. LVGL supporta un'ampia varietà di dispositivi di input utente:

- touch-screen,
- touch-pad,
- mouse,
- crown [scorimenti],
- encoder,

- keypad,
- keyboard,
- ecc.

Consultare [Creating an Input Device](#) per vedere come fare.

## API

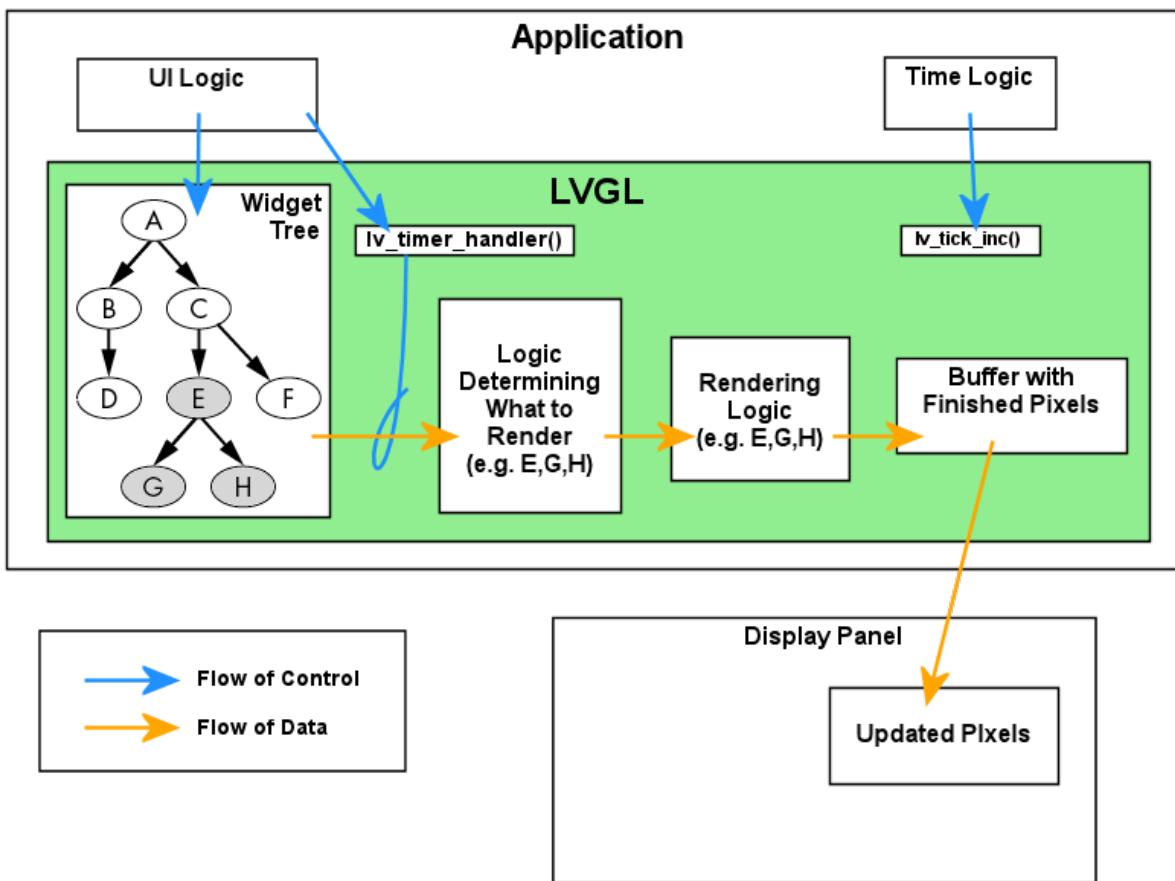
*lv\_tick.h*

*lv\_tick\_private.h*

### 4.1.5 Gestione del Timer

Per gestire i timer di LVGL è necessario chiamare periodicamente `lv_timer_handler()` in uno dei seguenti modi:

- `while(1)` della funzione `main()`, o
- periodicamente, un task del sistema operativo. (Vedere [LVGL e i Thread](#).)



Esempio:

```
while(1) {
    uint32_t time_till_next = lv_timer_handler();
    if(time_till_next == LV_NO_TIMER_READY) time_till_next = LV_DEF_REFR_PERIOD; /*handle LV_NO_TIMER_READY. Another option is to `sleep` for longer*/
    my_delay_ms(time_till_next);
}
```

Per utilizzare `lv_timer_handler()` in un super-loop, è disponibile la funzione helper `lv_timer_handler_run_in_period()` per semplificare l'impostazione di LVGL con la gestione del tempo:

```
while(1) {
...
lv_timer_handler_run_in_period(5); /* run lv_timer_handler() every 5ms */
...
}
```

Ottimamente utilizzare il tempo di sleep calcolato automaticamente da LVGL:

```
while(1) {
...
lv_timer_periodic_handler();
...
}
```

In un ambiente di sistema operativo, è possibile utilizzarlo insieme a **delay** o a **sleep** fornito dal sistema operativo per rilasciare la CPU quando possibile:

```
while (1) {
    uint32_t time_till_next = lv_timer_handler();
    if(time_till_next == LV_NO_TIMER_READY) time_till_next = LV_DEF_REFR_PERIOD; /*handle LV_NO_TIMER_READY. Another option is to `sleep` for longer*/
    os_delay_ms(time_till_next); /* delay to avoid unnecessary polling */
}
```

Leggere la sezione *Timer (lv\_timer)* per saperne di più sui timer.

## Quando Nessun Timer è Pronto

`lv_timer_handler()` restituirà **LV\_NO\_TIMER\_READY** (UINT32\_MAX) se non ci sono timer in esecuzione. Questo può accadere se non ci sono indevi o se sono disabilitati con `lv_indev_enable()`, animazioni in esecuzione o timer creati dall'utente in esecuzione. `lv_timer_handler()` continuerà a restituire **LV\_NO\_TIMER\_READY** finché non c'è un timer in esecuzione. I timer del display rimarranno in pausa quando non c'è motivo di aggiornarli. `lv_timer_handler()` dovrebbe essere chiamato dopo la creazione, l'eliminazione o la modifica di qualcosa in modo che venga eseguito un aggiornamento se necessario. In pratica, questo significa attendere senza timeout un evento esterno. Dopo che gli eventi esterni sono stati ricevuti e gestiti, `lv_timer_handler()` dovrebbe essere chiamato di nuovo.

```
while (1) {
    uint32_t time_till_next = lv_timer_handler();
    int timeout;

    /* Wait forever for events upon LV_NO_TIMER_READY, because there
     * is no reason to call lv_timer_handler sooner. */
    if(time_till_next == LV_NO_TIMER_READY) timeout = -1; /*infinite timeout*/

    /*Otherwise, wait for events at least until the timeout expires.*/
    else timeout = time_till_next;

    if(poll(..., timeout)) {
        /*Handle events before calling `lv_timer_handler` again.*/
    }
}
```

Se non c'è una sorgente di eventi esterni, si può scegliere di uscire dal ciclo o semplicemente ritardare a lungo.

Se un altro *thread* sta chiamando funzioni LVGL, si potrebbe voler chiamare nuovamente `lv_timer_handler()` molto presto per gestire gli effetti di quegli altri thread.

## API

*lv\_timer.h*

### 4.1.6 Considerazioni sui Thread

#### Definizioni

##### Thread

Nel contesto di questo documento, un *thread* è una qualsiasi sequenza di istruzioni della CPU. Nelle implementazioni "bare-metal" (ovvero senza sistema operativo), i *thread* includono:

- il *thread* principale che esegue un ciclo `while(1)` che esegue il sistema e
- Le interrupt service routine (ISR).

Quando vengono eseguiti su un sistema operativo, i *thread* includono:

- ogni task (o processo),

- ISR, e
- i sistemi operativi avanzati possono avere più "thread in esecuzione" all'interno di un processo.

### Operazione Atomica

Se l'operazione X è atomica, significa che qualsiasi thread che la osserva la vedrà come non ancora iniziata o come completata, e non in uno stato parzialmente completato.

Se altri thread possono vedere l'operazione in uno stato parzialmente eseguito o interferire con essa, allora l'operazione X non è atomica.

Se un'operazione atomica può fallire, la sua implementazione deve riportare la risorsa allo stato precedente all'avvio dell'operazione. Agli altri thread, deve apparire come se l'operazione non fosse ancora iniziata.

### Dati Atomici

Un dato (ovvero il contenuto di una variabile o di una struttura dati) è atomico se qualsiasi thread che lo osserva lo vedrà sempre in uno stato coerente, come se le operazioni su di esso non fossero ancora iniziate o fossero state completate con successo, e non in uno stato parzialmente modificato o altrimenti incoerente.

Quando la lettura o la scrittura di un valore viene avviata e completata con 1 istruzione CPU, è automaticamente atomica, poiché non può mai essere vista in uno stato incoerente (parzialmente modificato), nemmeno a causa di un'interruzione o di un'eccezione CPU. Con tali valori, i programmatore non richiedono alcuna protezione speciale per garantire che tutti i thread li vedano in uno stato coerente.

### LVGL e i Thread

#### LVGL non è thread-safe.

Ciò significa che è responsabilità del programmatore assicurarsi che nessuna funzione LVGL venga chiamata mentre è in corso un'altra chiamata LVGL in un altro thread. Questo include le chiamate a `lv_timer_handler()`.

#### Nota

Supponendo che quanto sopra sia il caso, è sicuro chiamare le funzioni LVGL nelle

- *callback di eventi* e nelle
- *callback del timer*

perché il thread che gestisce entrambi è il thread che chiama `lv_timer_handler()`.

Motivo:

LVGL gestisce molte strutture dati complesse, che sono "risorse di sistema" che devono essere protette dall'essere "viste" da altri thread in uno stato incoerente. Un'alta percentuale di funzioni LVGL (funzioni che iniziano con `lv_`) leggono o modificano tali strutture dati. Quelle che le modificano pongono i dati in uno stato incoerente durante l'esecuzione (poiché tali modifiche sono sequenze multi-step), ma li riportano a uno stato coerente prima che tali funzioni terminino. Per questo motivo, l'esecuzione di ciascuna funzione LVGL deve essere completata prima che venga avviata qualsiasi altra funzione LVGL.

#### Eccezioni a quanto Sopra

Queste due funzioni LVGL possono essere chiamate da qualsiasi thread:

- `lv_tick_inc()` (se la scrittura su `uint32_t` è atomica sulla piattaforma; vedere *Tick Interface* per maggiori informazioni) e
- `lv_display_flush_ready()` (*Flush Callback* per maggiori informazioni)

Il motivo per cui questo è accettabile è che i dati LVGL da loro modificati sono *atomici*.

Se un interrupt DEVE trasmettere informazioni a una parte dell'applicazione che chiama funzioni LVGL, imposta un flag o un altro valore atomico che il thread che chiama LVGL (o un *Timer LVGL* che si crea) possa leggere ed eseguire azioni.

Se si sta utilizzando un sistema operativo, ci sono alcune altre opzioni. Vedere di seguito.

## Garantire che gli Aggiornamenti Temporali Siano Atomici

Affinché i task LVGL relativi al tempo siano affidabili, gli aggiornamenti temporali tramite l'interfaccia Tick devono essere affidabili e il valore di Tick deve apparire *atomico* a LVGL. *Tick Interface* per i dettagli.

### Task

In un sistema operativo, è comune avere molti thread di esecuzione ("task" in alcuni sistemi operativi) che eseguono servizi per l'applicazione. In alcuni casi, tali thread possono acquisire dati che dovrebbero essere visualizzati (o altrimenti riflessi) nell'interfaccia utente, e ciò richiede chiamate LVGL per visualizzare tali dati (o modifiche).

Tuttavia, rimane responsabilità del programmatore assicurarsi che nessuna funzione LVGL venga chiamata mentre è in corso un'altra chiamata LVGL.

Come si fa?

### Metodo 1: Utilizzare un Gateway Thread

Un "Gateway Thread" (o "Gateway Task" in alcuni sistemi operativi) è un thread (task) che il progettista del sistema designa per gestire *esclusivamente* una risorsa di sistema. Un esempio è la gestione di un chip remoto, come una EEPROM o un altro dispositivo, che deve sempre essere portato in uno stato coerente prima dell'avvio di qualcosa di nuovo. Un altro esempio è la gestione di più dispositivi su un bus I2C (o qualsiasi bus dati). In questo caso, il bus I2C è la "risorsa gestita in modo esclusivo" e avere un solo thread che lo gestisce garantisce che ogni azione avviata possa essere completata prima che venga avviata un'altra azione con essa.

Le strutture dati di LVGL sono una risorsa di sistema che richiede tale protezione.

Utilizzando questo metodo, la creazione, la modifica e l'eliminazione di tutti i widget e delle altre risorse LVGL (ovvero tutte le chiamate di funzione LVGL, escluse le *eccezioni* menzionate in precedenza) vengono chiamate da quel thread. Ciò significa che il thread è anche l'UNICO chiamante di *lv\_timer\_handler()*. (*Aggiungere LVGL al Proprio Progetto* per ulteriori informazioni).

Questo garantisce che le strutture dati di LVGL "appaiano" *atomiche* (tutti i thread che utilizzano questi dati li "vedono" in uno stato coerente) perché nessun altro thread "visualizza" tali strutture dati. Questo è imposto dalla disciplina del programmatore che garantisce che il *Gateway Thread* sia l'unico thread a effettuare chiamate LVGL (escluse le *exceptions* menzionate sopra).

Se i *dati atomici* relativi all'interfaccia utente vengono aggiornati in un altro thread (ad esempio da un altro task o in un interrupt), il thread che chiama le funzioni LVGL può leggere tali dati direttamente senza preoccuparsi che siano in uno stato incoerente. (Per evitare un inutile sovraccarico della CPU, è possibile fornire un meccanismo [ad esempio, un flag generato dal thread di aggiornamento] in modo che l'interfaccia utente venga aggiornata solo quando ciò comporterà una modifica visibile all'utente finale).

Se i *dati non atomici* relativi all'interfaccia utente vengono aggiornati in un altro thread (ad esempio, da un altro task o durante un'interruzione), un modo alternativo (e sicuro) per trasmettere tali dati al thread che chiama le funzioni LVGL consiste nel passare una copia privata di tali dati a tale thread tramite una coda o un altro meccanismo del sistema operativo che protegga tali dati dalla visualizzazione in uno stato incoerente.

L'utilizzo di un *Gateway Thread* evita il sovraccarico di CPU (e di codice) derivante dall'utilizzo di un MUTEX per proteggere le strutture dati LVGL.

### Metodo 2: Utilizzare un MUTEX

Un MUTEX sta per "MUTually EXclusive" (reciprocamente esclusivo) ed è una primitiva di sincronizzazione che protegge lo stato di una risorsa di sistema da modifiche o accessi simultanei da parte di più thread di esecuzione. In altre parole, fa sì che i dati così protetti "appaiano" atomici (tutti i thread che utilizzano questi dati li "vedono" in uno stato coerente). La maggior parte dei sistemi operativi fornisce MUTEX.

Il progettista del sistema assegna un singolo MUTEX per produrre una singola risorsa di sistema. Una volta assegnato, il MUTEX esegue tale protezione da parte dei programmatori:

1. acquisendo il MUTEX (ovvero bloccandolo) prima di accedere o modificare tale risorsa, e
2. rilasciando il MUTEX (ovvero sbloccandolo) al termine dell'accesso o della modifica.

Se un thread tenta di acquisire (lock) il MUTEX mentre un altro thread ne è "proprietario", quel thread attende che l'altro thread lo rilasci (unlock) prima di poter continuare l'esecuzione.

Per essere chiari: questo deve essere fatto *sia* dai thread che LEGGONO da quella risorsa, sia dai thread che MODIFICANO quella risorsa.

Se un MUTEX viene utilizzato per proteggere le strutture dati LVGL, ciò significa *ogni* chiamata di funzione LVGL (o gruppo di chiamate di funzione) deve essere preceduta da #1 e seguita da #2, incluse le chiamate a `lv_timer_handler()`.

### Nota

Se il sistema operativo è integrato con LVGL (la macro `LV_USE_OS` ha un valore diverso da `LV_OS_NONE` in `lv_conf.h`) si possono usare `lv_lock()` e `lv_unlock()` per eseguire i punti #1 e #2.

In questo caso, `lv_timer_handler()` chiama internamente `lv_lock()` e `lv_unlock()`, quindi non si devono includere le chiamate a `lv_timer_handler()` tra di esse.

Se il sistema operativo NON è integrato con LVGL, queste chiamate o terminano immediatamente senza alcun effetto, oppure vengono ottimizzate dal linker.

Per abilitare `lv_lock()` e `lv_unlock()`, imposta `LV_USE_OS` su un valore diverso da `LV_OS_NONE`.

Questo pseudocodice illustra il concetto di utilizzo di un MUTEX:

```
void lvgl_thread(void)
{
    while(1) {
        uint32_t time_till_next;
        time_till_next = lv_timer_handler(); /* lv_lock/lv_unlock is called internally */
        if(time_till_next == LV_NO_TIMER_READY) time_till_next = LV_DEF_REFR_PERIOD; /*try again soon because the other thread can make the timer */
        →ready*/
        thread_sleep(time_till_next); /* sleep for a while */
    }
}

void other_thread(void)
{
    /* You must always hold (lock) the MUTEX while calling LVGL functions. */
    lv_lock();
    lv_obj_t *img = lv_image_create(lv_screen_active());
    lv_unlock();

    while(1) {
        lv_lock();
        /* Change to next image. */
        lv_image_set_src(img, next_image);
        lv_unlock();
        thread_sleep(2000);
    }
}
```

### Gestione della Modalità Sleep

L'MCU può entrare in modalità sleep quando non riceve input dall'utente per un certo periodo di tempo. In questo caso, il `while(1)` principale potrebbe apparire così:

```
while(1) {
    /* Normal operation (no sleep) in < 1 sec inactivity */
    if(lv_display_get_inactive_time(NULL) < 1000) {
        lv_timer_handler();
    }
    /* Sleep after 1 sec inactivity */
    else {
        timer_stop(); /* Stop the timer where lv_tick_inc() is called */
        sleep(); /* Sleep the MCU */
    }
    my_delay_ms(5);
}
```

Si devono anche aggiungere le seguenti righe alla funzione di lettura del dispositivo di input per segnalare l'avvenuta riattivazione (pressione, tocco, clic, ecc.):

```
lv_tick_inc(LV_DEF_REFR_PERIOD); /* Force task execution on wake-up */
timer_start(); /* Restart timer where lv_tick_inc() is called */
lv_timer_handler(); /* Call `lv_timer_handler()` manually to process the wake-up event */
```

Oltre a `lv_display_get_inactive_time()` si può controllare `lv_anim_count_running()` per verificare se tutte le animazioni sono terminate.

## 4.1.7 Altre Piattaforme e Tool

[Integrazione](#) per vedere come utilizzare LVGL su diverse piattaforme. Lì si troveranno molte descrizioni specifiche per ogni piattaforma, ad esempio per ESP32, Arduino, NXP, RT-Thread, NuttX, ecc.

# 4.2 Binding

## 4.2.1 API di Output come Dati JSON

A partire dal 20 giugno 2024, LVGL viene fornito con uno script Python (`./scripts/gen_json/gen_json.py`) che legge i file di header in LVGL e restituisce un formato JSON più intuitivo per l'API. Questo viene fatto in modo che i binding che generano codice automaticamente possano raccogliere facilmente le informazioni necessarie senza dover reinventare la ruota. Il formato JSON è stato scelto perché esistono librerie per la lettura di dati JSON in quasi tutti i linguaggi di programmazione.

### Requisiti

- Python >= 3.10
- Pyparser >= 2.22: Libreria Python per leggere l'output del preprocessore C
- PyMSVC >= 0.4.0: Libreria Python per l'utilizzo del compilatore MSVC
- Un compilatore C: gcc per Linux, clang per OSX e MSVC per Windows
- Doxygen: utilizzato per leggere i commenti di Doxygen (la documentazione dell'API) dai file di intestazione.

### Utilizzo

#### Opzioni della Riga di Comando

- `--output-path`: directory di output per il file JSON. Se non viene specificato, l'output verrà visualizzato su stdout.
- `--lvgl-config`: path per lv\_conf.h (incluso il nome del file). Se non è impostato, verrà generato un file di configurazione con le opzioni LVGL più comuni attivate.
- `--develop`: lascia i file generati nella cartella temporanea.

### Esempi

Utilizzo normale:

```
python ./scripts/gen_json/gen_json.py --output-path=json/output/directory --lvgl-config=path/to/lv_conf.h
```

Per eseguire un sottoprocesso dall'interno di uno script di generazione e leggere l'output da stdout:

```
python ./scripts/gen_json/gen_json.py --lvgl-config=path/to/lv_conf.h
```

### Dati di Output

Il contenuto del file di output è un oggetto JSON di grandi dimensioni (`{ . . . }`) con le seguenti coppie chiave/valore (queste sono le chiavi):

```
{
    "enums"          : [...],
    "functions"     : [...],
    "function_pointers": [...],
    "structures"    : [...],
    "unions"         : [...],
    "variables"      : [...],
    "typedefs"       : [...],
    "forward_decls" : [...],
    "macros"         : [...]
}
```

Come si vede, il valore di ciascuno di questi elementi è un array. Gli elementi di ogni array sono oggetti JSON, ciascuno con una struttura univoca per il tipo indicato dal nome dell'elemento genitore (ad esempio "enum", "funzioni", ecc.).

A ciascun oggetto è stata aggiunta una coppia chiave/valore (chiave = "json\_type") per consentire il passaggio di un oggetto a una funzione generica e far sì che ciascun oggetto conosca il proprio tipo tramite questo campo. I possibili valori di "json\_type" sono:

- **"array"**: Il tipo dell'array viene utilizzato per identificare gli array.

Campi:

- **"dim"**: numero di elementi nell'array
- **"quals"**: array di qualificatori, ad esempio, "const"
- **"type"**: Questo potrebbe essere disponibile o meno.
- **"name"**: nome del tipo di dati

- **"field"**: Questo tipo viene utilizzato per descrivere i campi in strutture e unioni. Viene utilizzato nell'array "fields" dei tipi **"struct"** **"union"** trattati di seguito.

Campi:

- **"name"**: nome del campo
- **"type"**: data type
- **"json\_type"** carries object type (e.g. "enum", "function", etc.) identifying the top-level group it comes from
- **"bitsize"**: Numero di bit per i campi bit, o **null** per i tipi di campo normali.
- **"docstring"**: stringa contenente la documentazione estratta da Doxygen.

- **"arg"**: Descrive un argomento di funzione.

Campi:

- **"name"**: nome dell'argomento
- **"type"**: data type
- **"json\_type"** carries object type (e.g. "enum", "function", etc.) identifying the top-level group it comes from.
- **"docstring"**: stringa contenente la documentazione estratta da Doxygen.
- **"quals"**: array di tutti i qualificatori presenti, ad esempio "const"

- **"forward\_decl"**: Descrive una dichiarazione in avanti (forward). In LVGL ci sono strutture considerate private, ed è proprio ciò che descrivono.

Campi:

- **"name"**: nome della dichiarazione forward
- **"type"**: data type
- **"json\_type"** carries object type (e.g. "enum", "function", etc.) identifying the top-level group it comes from.
- **"docstring"**: stringa contenente la documentazione estratta da Doxygen.
- **"quals"**: array di tutti i qualificatori presenti, ad esempio "const"

- **"function\_pointer"**: Descrive un puntatore a funzione. Questi vengono utilizzati per la registrazione di funzioni di callback in LVGL.

Campi:

- **"name"**: nome del puntatore a funzione

- "type": tipo di ritorno della funzione
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "args": array degli oggetti "arg" descritti sopra
- "quals": array di tutti i qualificatori presenti, ad esempio "const"
- "variable": Descrive una variabile globale.

Campi:

- "name": nome della variabile
- "type": data type
- "json\_type" carries object type (e.g. "enum", "function", etc.) identifying the top-level group it comes from.
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "quals": array di tutti i qualificatori presenti, ad esempio "const"
- "storage": array di tutti gli specificatori di classe di archiviazione presenti (ad esempio "auto", "static", "extern", ecc.)
- "special\_type": Attualmente utilizzato solo per descrivere un argomento con ellissi di una funzione.

Campi:

- "name": sempre "ellissi"
- "primitive\_type": Tipo di dati che non inizia con "lv\_" e non termina con "\_t". Confrontare con "lvgl\_type". Include struct, union, tipi interi (ad esempio int, unsigned int), ecc.

Campi:

- "name": nome del tipo primitivo
- "enum": Enumerazioni C

Campi:

- "name": Se l'enumerazione è il risultato di un **typedef**, questo campo riporta il nome del tipo definito. Esempio: **lv\_align\_t**. (Non sempre disponibile).
- "type": tipo di enumeratori (sempre "int")
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "members": array di oggetti "enum\_member"
- "enum\_member": enumeratore (valore di enumerazione). Questo "json\_type" si trova solo nell'array "members" di un oggetto "enum"

Campi:

- "name": nome dell'enumeratore
- "type": Se l'enumerazione è il risultato di un **typedef**, questo campo contiene il nome del tipo definito. Esempio: **lv\_align\_t**.
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "value": valore dell'enumeratore
- "lvgl\_type": Tipo di dati definito in LVGL (inizia con "lv\_" e termina con "\_t").

Campi:

- "name": nome del tipo
- "quals": array di tutti i qualificatori presenti, ad esempio "const"

- "struct": struct C

Campi:

- "name": nome della struct (tipo di dati se definito da `typedef`)
- "type": un oggetto "primitive\_type" object {"name": "struct", "json\_type": "primitive\_type"}. (Vedere la definizione sopra).
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "fields": array di oggetti "field" (Vedere la definizione sopra).
- "quals": array di tutti i qualificatori presenti, ad esempio "const"

- "union": union C

Campi:

- "name": nome della union (tipo di dati se definito da `typedef`)
- "type": un oggetto "primitive\_type" {"name": "union", "json\_type": "primitive\_type"}. (Vedere la definizione sopra).
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "fields": array di elementi "field".
- "quals": array di tutti i qualificatori presenti, ad esempio "const"

- "macro": macro C. Le informazioni che si possono raccogliere sulle macro sono limitate e nella maggior parte dei casi un binding richiederà che queste vengano aggiunte staticamente a un binding. Serve più per raccogliere le docstring che altro.

Campi:

- "name": nome della macro
- "docstring": stringa contenente la documentazione estratta da Doxygen.

- "ret\_type": tipo di ritorno da una funzione. Questo verrà visualizzato solo nell'elemento "type" di un tipo "function".

Campi:

- "type": data type
- "json\_type" carries object type (e.g. "enum", "function", etc.) identifying the top-level group it comes from.
- "docstring": stringa contenente la documentazione estratta da Doxygen.

- "function": funzione C

Campi:

- "name": nome della funzione
- "type": Un oggetto "ret\_type". (Vedere la definizione sopra).
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "args": array of "arg" tipi json. (Vedere la definizione sopra).

- "stdlib\_type": tipo intero C (int, unsigned int, float, ecc.)

Campi:

- "name": nome del tipo
- "quals": array di tutti i qualificatori presenti, ad esempio "const"

- "unknown\_type": Questo non dovrebbe essere visibile. In tal caso, è necessario apportare una modifica allo script. Aprire una segnalazione per farci sapere se si vede questo tipo.

Campi:

- "name": nome del tipo
- "quals": array di tutti i qualificatori presenti, ad esempio "const"

- "pointer": puntatore C

Campi:

- "type": pointer type
- "json\_type" carries object type (e.g. "enum", "function", etc.) identifying the top-level group it comes from.
- "quals": array di tutti i qualificatori presenti, ad esempio "const"

- "typedef": definizione del tipo C

Campi:

- "name": nome del tipo (ad esempio `lv_part_t`)
- "type": un oggetto "primitive\_type" {"name": "uint32\_t", "json\_type": "stdlib\_type"}. (Vedere la definizione sopra).
- "json\_type" carries object type (e.g. "enum", "function", etc.) identifying the top-level group it comes from.
- "docstring": stringa contenente la documentazione estratta da Doxygen.
- "quals": array di tutti i qualificatori presenti, ad esempio "const"

Ecco un esempio abbreviato di come appare l'output.

```
{
  "enums": [
    {
      "name": "_lv_result_t",
      "type": {
        "name": "int",
        "json_type": "primitive_type"
      },
      "json_type": "enum",
      "docstring": "LVGL error codes. ",
      "members": [
        {
          "name": "LV_RESULT_INVALID",
          "type": {
            "name": "_lv_result_t",
            "json_type": "lvgl_type"
          },
          "json_type": "enum_member",
          "docstring": "",
          "value": "0x0"
        },
        {
          "name": "LV_RESULT_OK",
          "type": {
            "name": "_lv_result_t",
            "json_type": "lvgl_type"
          },
          "json_type": "enum_member",
          "docstring": "",
          "value": "0x1"
        }
      ]
    }
  ],
  "functions": [
    {
      "name": "lv_version_info",
      "type": {
        "type": {
          "type": {
            "name": "char",
            "json_type": "primitive_type",
            "quals": [
              "const"
            ]
          },
          "json_type": "pointer",
          "quals": []
        },
        "json_type": "ret_type",
        "docstring": ""
      },
      "json_type": "function",
      "docstring": "",
      "args": [
        ...
      ]
    }
  ]
}
```

(continues on next page)

(continua dalla pagina precedente)

```
{
    "name": "null",
    "type": {
        "name": "void",
        "json_type": "primitive_type",
        "quals": []
    },
    "json_type": "arg",
    "docstring": "",
    "quals": []
}
]
},
"function_pointers": [
{
    "name": "lv_tlsf_walker",
    "type": {
        "type": {
            "name": "void",
            "json_type": "primitive_type",
            "quals": []
        },
        "json_type": "ret_type",
        "docstring": ""
    },
    "json_type": "function_pointer",
    "docstring": "",
    "args": [
        {
            "name": "ptr",
            "type": {
                "type": {
                    "name": "void",
                    "json_type": "primitive_type",
                    "quals": []
                },
                "json_type": "pointer",
                "quals": []
            },
            "json_type": "arg",
            "docstring": ""
        },
        {
            "name": "size",
            "type": {
                "name": "size_t",
                "json_type": "stdlib_type",
                "quals": []
            },
            "json_type": "arg",
            "docstring": ""
        },
        {
            "name": "used",
            "type": {
                "name": "int",
                "json_type": "primitive_type",
                "quals": []
            },
            "json_type": "arg",
            "docstring": ""
        },
        {
            "name": "user",
            "type": {
                "type": {
                    "name": "void",
                    "json_type": "primitive_type",
                    "quals": []
                },
                "json_type": "pointer",
                "quals": []
            },
            "json_type": "arg",
            "docstring": ""
        }
    ],
    "quals": []
},
{
    "name": "_lv_grad_cache_t",
    "type": {
        "name": "struct",
        "json_type": "primitive_type"
    },
    "json_type": "struct",
    "docstring": null,
    "fields": [
        {
            "name": "color_map",
            "type": {
                "type": {
                    "name": "lv_color_t",
                    "json_type": "lvgl_type",
                    "quals": []
                },
                "json_type": "pointer",
                "quals": []
            },
            "json_type": "field",
            "bitsize": null,
            "docstring": ""
        },
        {
            "name": "opa_map",
            "type": {
                "type": {
                    "name": "lv_opa_t",
                    "json_type": "lvgl_type",
                    "quals": []
                },
                "json_type": "pointer",
                "quals": []
            },
            "json_type": "field",
            "bitsize": null,
            "docstring": ""
        }
    ]
}
],
"structures": [
{
    "name": "_lv_grad_cache_t",
    "type": {
        "name": "struct",
        "json_type": "primitive_type"
    },
    "json_type": "struct",
    "docstring": null,
    "fields": [
        {
            "name": "color_map",
            "type": {
                "type": {
                    "name": "lv_color_t",
                    "json_type": "lvgl_type",
                    "quals": []
                },
                "json_type": "pointer",
                "quals": []
            },
            "json_type": "field",
            "bitsize": null,
            "docstring": ""
        },
        {
            "name": "opa_map",
            "type": {
                "type": {
                    "name": "lv_opa_t",
                    "json_type": "lvgl_type",
                    "quals": []
                },
                "json_type": "pointer",
                "quals": []
            },
            "json_type": "field",
            "bitsize": null,
            "docstring": ""
        }
    ]
}
]
```

(continues on next page)

(continua dalla pagina precedente)

```

        "quals": []
    },
    "json_type": "pointer",
    "quals": []
},
{
    "name": "size",
    "type": {
        "name": "uint32_t",
        "json_type": "stdlib_type",
        "quals": []
    },
    "json_type": "field",
    "bitsize": null,
    "docstring": ""
}
]
],
"unions": [],
"variables": [
{
    "name": "lv_global",
    "type": {
        "name": "lv_global_t",
        "json_type": "lvgl_type",
        "quals": []
    },
    "json_type": "variable",
    "docstring": "",
    "quals": [],
    "storage": [
        "extern"
    ]
}
],
"typedefs": [
{
    "name": "lv_pool_t",
    "type": {
        "type": {
            "name": "void",
            "json_type": "primitive_type",
            "quals": []
        },
        "json_type": "pointer"
    },
    "json_type": "typedef",
    "docstring": "",
    "quals": []
}
],
"forward_decls": [
{
    "name": "lv_fragment_managed_states_t",
    "type": {
        "name": "struct",
        "json_type": "primitive_type"
    },
    "json_type": "forward_decl",
    "docstring": "",
    "quals": []
}
],
"macros": [
{
    "name": "ZERO_MEM_SENTINEL",
    "json_type": "macro",
    "docstring": ""
}
]
}
}

```

## 4.2.2 Cpp

In progress: [https://github.com/lvgl/lv\\_binding\\_cpp](https://github.com/lvgl/lv_binding_cpp)

## 4.2.3 JavaScript

Con `lv_binding_js` si può usare LVGL dall'interno di JavaScript.

Sfrutta il concetto di DOM virtuale di React per manipolare i componenti dell'interfaccia utente LVGL, offrendo agli utenti un'esperienza familiare simile a React.

### Codice

#### Codice in Esecuzione su Dispositivo Reale

## Funzionalità

- Supporta tutti i componenti LVGL nativi
- Supporta pienamente gli stili flex e grid LVGL
- Supporta la maggior parte degli stili LVGL, basta scrivere come CSS HTML5
- Supporta il caricamento dinamico delle immagini
- Supporta pienamente le animazioni LVGL

## Demo

Vedere la cartella [demo](#)

## Building

Di seguito sono riportate alcune note per sviluppatori su come compilare lvgljs sulla piattaforma nativa. Non si tratta di guide complete, ma includono note sulle librerie necessarie, i flag di compilazione, ecc.

## lvgljs

- Note di build per dispositivi Linux embedded
- Note di build per SDL Simulator (Linux e macOS)

## Bundle JS

- Note di build del bundle JS

## Components

- View
- Image
- Button
- Text
- Input
- Textarea
- Switch
- Checkbox
- Dropdownlist
- ProgressBar
- Line
- Roller
- Keyboard
- Calendar
- Chart

## Font

- [Builtin-Symbol](#)

## Animation

- Animation

## Style

- position-size-layout
- boxing-model
- color
- flex
- grid
- font
- opacity
- display
- background
- scroll
- shadow
- recolor
- line
- transition
- transform

## JS API

- network
- filesystem
- dimension

## Thanks

**lvgl.js** depends on following excellent work:

- [lvgl](#): Create beautiful UIs for any MCU, MPU and display type
- [QuickJS](#): JavaScript engine
- [libuv](#): platform abstraction layer
- [curl](#): HTTP client
- [txiki.js](#): Tiny JavaScript runtime

## 4.2.4 MicroPython

### Cos'è MicroPython?

MicroPython è Python per microcontrollori. Utilizzando MicroPython, è possibile scrivere codice Python3 ed eseguirlo anche su un'architettura bare metal con risorse limitate. Una delle sue potenti funzionalità è la possibilità di modificare il comportamento di un dispositivo modificando il codice Python su una memoria rimovibile (o interna), senza dover modificare il firmware del dispositivo.

## Punti di Forza di MicroPython

### Compatto

Si adatta e funziona in soli 256 kB di spazio di codice e 16 kB di RAM. Non è necessario alcun sistema operativo, anche se è possibile eseguirlo anche con un sistema operativo, volendo.

### Compatibile

Cerca di essere il più compatibile possibile con Python standard (noto come CPython).

### Versatile

Supporta molte architetture (x86, x86-64, ARM, ARM Thumb, Xtensa).

### Interattivo

Non è necessario il ciclo di compilazione-flash-boot. Con REPL (prompt interattivo) è possibile digitare comandi ed eseguirli immediatamente, eseguire script, ecc.

### Popolare

Sono supportate numerose piattaforme. La base utenti è in continua crescita. Fork degne di nota:

- [MicroPython](#)
- [CircuitPython](#)
- [MicroPython\\_ESP32\\_psRAM\\_LoBo](#)

### Orientato agli Embedded

Fornito con moduli specifici per sistemi embedded, come il [modulo machine](#) per l'accesso all'hardware di basso livello (pin I/O, ADC, UART, SPI, I2C, RTC, timer ecc.)

## Perché MicroPython + LVGL?

MicroPython non dispone di una buona libreria GUI nativa di alto livello. LVGL è una libreria GUI di alto livello Basata su [Componenti Object-Oriented](#) che è una candidata naturale per essere mappata in un linguaggio di livello superiore, come Python. LVGL è implementato in C e le sue API sono in C.

## Ecco alcuni vantaggi dell'utilizzo di LVGL in MicroPython:

- Sviluppo di GUI in Python, un linguaggio di alto livello molto diffuso. Utilizzo di paradigmi come la Programmazione Object-Oriented.
- Di solito, lo sviluppo di GUI richiede più iterazioni per ottenere risultati ottimali. In C, ogni iterazione consiste in ``**Modifica del codice**`` > ``**Build**`` > ``**Flash**`` > ``**Run**``. In MicroPython è semplicemente ``**Modifica del codice**`` > ``**Run**`` ! È anche possibile eseguire comandi in modo interattivo utilizzando [REPL](#) (il prompt interattivo)

## MicroPython + LVGL potrebbe essere utilizzato per:

- Prototipazione rapida di GUI
- Ridurre il ciclo di modifica e messa a punto di una GUI
- Modellare una GUI in modo più astratto definendo widget compositi riutilizzabili, sfruttando le funzionalità del linguaggio Python come ereditarietà, chiusure, comprensione di liste, generatori, gestione delle eccezioni, interi a precisione arbitraria e altre.
- Rendere LVGL accessibile a un pubblico più vasto. Non è necessario conoscere il C per creare una buona GUI su un sistema embedded. Questo si integra perfettamente con la [visione CircuitPython](#). CircuitPython è stato progettato pensando alla formazione, per facilitare l'approccio allo sviluppo embedded da parte di programmatore nuovi o inesperti.
- Creare strumenti per lavorare con LVGL a un livello superiore (ad esempio, un designer drag-and-drop).

## Che aspetto ha?

È molto simile all'API C, ma orientata agli oggetti per i componenti LVGL.

Immergiamoci subito in un esempio!

## Un semplice esempio

```
# Initialize
import display_driver
import lvgl as lv

# Create a button with a label
scr = lv.obj()
btn = lv.button(scr)
btn.align(lv.ALIGN.CENTER, 0, 0)
label = lv.label(btn)
label.set_text('Hello World!')
lv.screen_load(scr)
```

## Come si Usa?

### Simulatore Online

Per sperimentare con LVGL + MicroPython senza scaricare nulla, si può usare il simulatore online! È un simulatore LVGL + MicroPython completamente funzionante che funziona interamente nel browser e permette di modificare uno script Python ed eseguirlo.

[Cliccare qui per sperimentare sul simulatore online](#)

Molti *esempi LVGL* sono disponibili anche per MicroPython. Basta cliccare sul link!

### Simulatore per PC

MicroPython è disponibile su numerose piattaforme. Una versione degna di nota è quella per "Unix", che consente di compilare ed eseguire MicroPython (+LVGL) su una macchina Linux. (Su una macchina Windows potrebbe essere necessario Virtual Box, WSL, MinGW, Cygwin, ecc.)

[Cliccare qui per saperne di più sulla compilazione e l'esecuzione della versione Unix.](#)

### Piattaforme Embedded

In definitiva, l'obiettivo è eseguire il tutto su una piattaforma embedded. Sia MicroPython che LVGL possono essere utilizzati su numerose architetture embedded. [lv\\_micropython](#) è un fork di MicroPython+LVGL e attualmente supporta Linux, ESP32, STM32 e RP2. Può essere portato su qualsiasi altra piattaforma supportata da MicroPython.

- Sono necessari anche driver di input e display. È possibile utilizzare uno dei driver esistenti forniti con [lv\\_micropython](#) oppure creare driver di input/display personalizzati per l'hardware specifico.
- I driver possono essere implementati in C come modulo MicroPython o in Python puro.

**lv\_micropython** contiene già i seguenti driver:

- Driver per display:
  - SDL su Linux
  - X11 su Linux
  - Specifici per ESP32:
    - \* ILI9341
    - \* ILI9488
    - \* GC9A01
    - \* ST7789
    - \* ST7735
  - Generici (Python puro):

- \* ILI9341
- \* ST7789
- \* ST7735
- Driver di input:
  - SDL
  - X11
  - XPT2046
  - FT6X36
  - ESP32 ADC con touch resistivo

### Dove si possono trovare maggiori informazioni?

- [lv\\_micropython README](#)
- [lv\\_binding\\_micropython README](#)
- Il forum [LVGL micropython](#) (Sentitevi liberi di chiedere qualsiasi cosa!)
- Su MicroPython: [docs](#) e [forum](#)
- Post del [Blog](#), un po' obsoleto.

### Il Binding MicroPython è generato automaticamente!

- LVGL è un sottomodulo git all'interno di [lv\\_micropython](#) (LVGL è un sottomodulo git di [lv\\_binding\\_micropython](#) che a sua volta è un sottomodulo di [lv\\_micropython](#)).
- Durante la compilazione di [lv\\_micropython](#), l'API C pubblica di LVGL viene analizzata e l'API MicroPython viene generata automaticamente. Ciò significa che [lv\\_micropython](#) fornisce l'API LVGL per **qualsiasi** versione di LVGL e generalmente non richiede modifiche al codice man mano che LVGL si evolve.

### Convenzioni di codifica dell'API C di LVGL

Per un riepilogo delle convenzioni di codifica da seguire, consultare [Stile di Codifica](#).

### Gestione della Memoria

- Quando LVGL viene eseguito in MicroPython, tutte le allocazioni di memoria dinamica ([lv\\_malloc\(\)](#)) vengono gestite dal gestore di memoria di MicroPython, che è **garbage-collected** (GC).
- Per evitare che il GC raccolga la memoria prematuramente, tutta la RAM allocata dinamicamente deve essere raggiungibile dal GC.
- Il GC riconosce la maggior parte delle allocazioni, ad eccezione dei puntatori al Data Segment:
  - Puntatori che sono variabili globali
  - Puntatori che sono variabili globali statiche
  - Puntatori che sono variabili locali statiche

Tali puntatori devono essere definiti in modo specifico per renderli raggiungibili dal GC.

### Identificare Il Problema

Un problema si verifica quando un puntatore di memoria allocata (valore di ritorno di [lv\\_malloc\(\)](#)) viene memorizzato solo in una variabile puntatore **global**, **static global** o **static local** e non come parte di una **struct** precedentemente allocata o di un'altra variabile.

## Risoluzione del Problema

- Sostituire la variabile locale globale/statica con (`LV_GLOBAL_DEFAULT() -> _var`)
- Includere `lv_global.h` nei file che usano `LV_GLOBAL_DEFAULT`
- Aggiungere `_var` a `lv_global_t` su `lv_global.h`

## Esempio

### Ulteriori Informazioni sulla Gestione della Memoria

- Nel [README](#)
- Nel [Blog](#)

## Callback

In C, un callback è semplicemente un puntatore a funzione. In MicroPython, invece, dobbiamo registrare un *oggetto richiamabile MicroPython* per ogni callback. Pertanto, nel binding MicroPython dobbiamo registrare sia un puntatore a funzione che un oggetto MicroPython per ogni callback.

Abbiamo quindi definito una **convenzione di callback** per l'API C LVGL che prevede che gli header LVGL siano definiti in un certo modo. Le callback dichiarate secondo questa convenzione consentono al binding di registrare un oggetto MicroPython accanto al puntatore a funzione quando si registra una callback e di accedere a tale oggetto quando la callback viene chiamata.

- L'idea di base è che abbiamo un campo `void * user_data` che viene utilizzato automaticamente dal binding MicroPython per salvare l'*oggetto richiamabile MicroPython* per una callback. Questo campo deve essere specificato durante la registrazione del puntatore alla funzione e fornito alla funzione di callback stessa.
- Sebbene chiamato "user\_data", non è previsto che l'utente legga/scriva tale campo. Invece, il codice "glue" di MicroPython usa `user_data` per tenere traccia automaticamente dell'oggetto richiamabile di MicroPython. Il codice "glue" lo aggiorna quando la callback viene registrata e lo utilizza quando la callback viene chiamata per invocare una chiamata all'oggetto richiamabile originale.

Esistono alcune opzioni per definire un callback nell'API C LVGL:

- Opzione 1: `user_data` in una struct
  - Esiste una struttura che contiene un campo chiamato `void * user_data`
    - \* Un puntatore a tale struttura viene fornito come **primo** argomento di una funzione di registrazione della callback.
    - \* Un puntatore a tale struttura viene fornito come **primo** argomento della callback stessa.
- Opzione 2: `user_data` come argomento della funzione
  - Un parametro chiamato `void * user_data` viene fornito alla funzione di registrazione come **ultimo** argomento
    - \* La callback stessa riceve `void *` come **ultimo** argomento
- Opzione 3: sia la callback che `user_data` sono campi di struct
  - L'API espone una struct con sia un membro puntatore a funzione che un membro `user_data`
    - \* Il membro puntatore a funzione riceve la stessa struct come suo **primo** argomento

In pratica è anche possibile combinare queste opzioni, ad esempio fornire un puntatore a struct quando si registra una callback (opzione 1) e fornire l'argomento `user_data` quando si chiama la callback (opzione 2), **purché lo stesso user\_data registrato venga passato alla callback quando viene chiamata**.

## Esempi

- `lv_anim_t` contiene il campo `user_data`. `lv_anim_set_path_cb()` registra la callback `path_cb`. Sia `lv_anim_set_path_cb` che `lv_anim_path_cb_t` ricevono `lv_anim_t` come primo argomento
- Il campo `path_cb` può anche essere assegnato direttamente nel codice Python perché è un membro di `lv_anim_t` che contiene il campo `user_data`, e `lv_anim_path_cb_t` riceve `lv_anim_t` come primo argomento.
- `lv_imgfont_create()` registra `path_cb` e riceve `user_data` come ultimo argomento. Anche la callback `lv_imgfont_get_path_cb_t` riceve `user_data` come ultimo argomento.

## Ulteriori Approfondimenti sulle Callback

- Nel [Blog](#) e nel [README](#)
- [v6.0] [Callback conventions #1036](#)
- Varie discussioni: [qui](#), [qui](#) e [qui](#)

## 4.2.5 PikaScript

### Cos'è PikaScript?

PikaScript è un interprete Python progettato specificamente per microcontrollori e supporta un sottoinsieme della comune sintassi Python3.

È più leggero di MicroPython, richiedendo solo 32k di spazio codice e 4k di RAM, il che significa che può essere eseguito su stm32f103c8 (blue-pill) o persino stm32g030c8. D'altra parte, è possibile lasciare spazio prezioso per più materiale o aree di buffer più grandi.

È più semplice, pronto all'uso, funziona senza alcun porting o configurazione, non dipende dal sistema operativo o dal file system, offre un buon supporto per gli IDE più diffusi per piattaforme Windows come Keil, IAR, RT-Thread-Studio e, naturalmente, supporta le piattaforme di sviluppo Linux gcc.

È più intelligente, con un meccanismo esclusivo per i moduli C che consente di generare automaticamente i binding semplicemente scrivendo l'API per il modulo C in Python, senza dover affrontare il problema di dover scrivere manualmente macro o tabelle globali. D'altra parte, tutti i moduli C dispongono di suggerimenti intelligenti sofisticati, che suggeriscono persino i tipi degli argomenti.

### Perché PikaScript + LVGL?

- PikaScript ora supporta le funzionalità principali di LVGL8 e queste API sono pienamente compatibili con MicroPython. Ciò significa che è possibile continuare a utilizzare il codice già scritto in MicroPython, ma occupando meno spazio di codice e RAM.
- Utilizzare suggerimenti dettagliati sul codice fino al tipo di parametro per una migliore esperienza di programmazione.
- Utilizzare un IDE più pratico, come i progetti di simulazione basati su Visual Studio.

### Come si Presenta?

Ecco alcuni esempi di utilizzo di LVGL che PikaScript può già eseguire. Provengono principalmente dagli esempi della documentazione di LVGL.

## LV\_ARC

```
import pika_lvgl as lv
import PikaStdLib
mem = PikaStdLib.MemChecker()
# Create an Arc
arc = lv.arc(lv.screen_active())
arc.set_end_angle(200)
arc.set_size(150, 150)
arc.center()
print('mem used max: %0.2f kB' % (mem.getMax()))
print('mem used now: %0.2f kB' % (mem.getNow()))
```

## LV\_BAR

```
import pika_lvgl as lv
import PikaStdLib
mem = PikaStdLib.MemChecker()
bar1 = lv.bar(lv.screen_active())
bar1.set_size(200, 20)
bar1.center()
bar1.set_value(70, lv.ANIM.OFF)
print('mem used max: %0.2f kB' % (mem.getMax()))
print('mem used now: %0.2f kB' % (mem.getNow()))
```

## LV\_BTN

```
import pika_lvgl as lv
import PikaStdLib
mem = PikaStdLib.MemChecker()

def event_cb_1(evt):
    print('in evt1')
    print('mem used now: %0.2f kB' % (mem.getNow()))

def event_cb_2(evt):
    print('in evt2')
    print('mem used now: %0.2f kB' % (mem.getNow()))

btn1 = lv.btn(lv.screen_active())
btn1.align(lv.ALIGN.TOP_MID, 0, 10)
btn2 = lv.btn(lv.screen_active())
btn2.align(lv.ALIGN.TOP_MID, 0, 50)
btn1.add_event_cb(event_cb_1, lv.EVENT.CLICKED, 0)
btn2.add_event_cb(event_cb_2, lv.EVENT.CLICKED, 0)
print('mem used max: %0.2f kB' % (mem.getMax()))
print('mem used now: %0.2f kB' % (mem.getNow()))
```

## LV\_CHECKBOX

```
import pika_lvgl as lv
import PikaStdLib
mem = PikaStdLib.MemChecker()
cb = lv.checkbox(lv.screen_active())
cb.set_text("Apple")
cb.align(lv.ALIGN.TOP_LEFT, 0, 0)
cb = lv.checkbox(lv.screen_active())
cb.set_text("Banana")
cb.add_state(lv.STATE.CHECKED)
cb.align(lv.ALIGN.TOP_LEFT, 0, 30)
cb = lv.checkbox(lv.screen_active())
cb.set_text("Lemon")
cb.add_state(lv.STATE.DISABLED)
cb.align(lv.ALIGN.TOP_LEFT, 0, 60)
cb = lv.checkbox(lv.screen_active())
cb.add_state(lv.STATE.CHECKED | lv.STATE.DISABLED)
cb.set_text("Melon")
cb.align(lv.ALIGN.TOP_LEFT, 0, 90)
print('mem used max: %0.2f kB' % (mem.getMax()))
print('mem used now: %0.2f kB' % (mem.getNow()))
```

## Come Funziona?

PikaScript dispone di un esclusivo strumento di binding intelligente per i moduli C.

Basta scrivere l'interfaccia Python in pika\_lvgl.pyi (.pyi è un file di interfaccia Python).

```
# pika_lvgl.pyi
class arc(lv_obj):
    def set_end_angle(self, angle: int): ...
    def set_bg_angles(self, start: int, end: int): ...
    def set_angles(self, start: int, end: int): ...
```

Il precompilatore di PikaScript può poi associare automaticamente le seguenti funzioni C, semplicemente nominando le funzioni nel formato modulo\_classe\_mетодо, senza alcun lavoro aggiuntivo, e tutto il binding e la registrazione vengono eseguiti automaticamente.

```
/* pika_lvgl_arc.c */
void pika_lvgl_arc_set_end_angle(PikaObj* self, int angle) {
    lv_obj_t* lv_obj = obj_getPtr(self, "lv_obj");
    lv_arc_set_end_angle(lv_obj, angle);
}
void pika_lvgl_arc_set_bg_angles(PikaObj *self, int start, int end){
    lv_obj_t* lv_obj = obj_getPtr(self, "lv_obj");
    lv_arc_set_bg_angles(lv_obj, start, end);
}
void pika_lvgl_arc_set_angles(PikaObj *self, int start, int end){
    lv_obj_t* lv_obj = obj_getPtr(self, "lv_obj");
    lv_arc_set_angles(lv_obj, start, end);
}
```

Per utilizzare il modulo, basta `import pika_lvgl` e il precompilatore analizzerà automaticamente `main.py` e associerà il modulo `pika_lvgl`.

```
$ ./rust-msc-latest-win10.exe
(pikascript) packages installed:
pikascript-core==v1.10.0
PikaStdLib==v1.10.0
PikaStdDevice==v1.10.0
(pikascript) pike compiler:
scanning main.py...
binding pika_lvgl.pyi...
```

Il precompilatore è scritto in Rust, funziona su Windows e Linux ed è completamente open source.

Oltre ad associare i moduli C, il precompilatore compila gli script Python in bytecode nel PC, riducendo le dimensioni dello script e aumentandone la velocità.

## Come si Usa?

Il repository di simulazione per Visual Studio è disponibile all'indirizzo [https://github.com/pikasTech/lv\\_pikascript](https://github.com/pikasTech/lv_pikascript)

## 4.3 Sistemi di Build

### 4.3.1 make

LVGL può essere facilmente integrato in qualsiasi progetto Makefile aggiungendo `include lvgl/lvgl.mk` al Makefile principale.

### 4.3.2 CMake

#### Panoramica

CMake è un generatore di sistemi di build multipiattaforma. Viene utilizzato per integrare facilmente un progetto/libreria in un altro progetto. Offre inoltre la possibilità di configurare la build con diverse opzioni, di abilitare o disabilitare componenti o di integrare l'esecuzione di script personalizzati durante la fase di configurazione/build.

LVGL include CMake in modo nativo, il che significa che è possibile utilizzarlo per configurare e compilare LVGL direttamente o integrarlo in una build CMake di livello superiore.

Questo progetto utilizza i preset CMake per garantire una build semplice. Maggiori informazioni su Cmake Presets qui: <https://cmake.org/cmake/help/latest/manual/cmake-presets.7.html>

#### Prerequisiti

È necessario installare

- CMake con GNU make o Ninja (per le build Linux). Aggiungere ninja/make al PATH!
- I prerequisiti sono elencati in `scripts/install-prerequisites.sh/bat`
- Un interprete Python3 per utilizzare KConfig.

## Come compilare questo progetto con cmake

### Compilazione da riga di comando

Il modo più semplice per compilare LVGL con cmake è usare le chiamate da riga di comando:

```
# Method 1
cd <lvgl_repo>
mkdir build
cd build
cmake ..           # Configure phase
cmake --build .    # Build phase

# Method 2
cd <lvgl_repo>
cmake -B build     # Configure phase
cmake --build build # build phase
```

### Compilazione con i preset di CMake

Un altro modo per compilare questo progetto è utilizzare il file CMakePresets.json fornito o passare le opzioni tramite la riga di comando. Il file CMakePresets.json descrive alcune configurazioni di CMake e la fase di compilazione. È un modo per utilizzare rapidamente un set di opzioni predefinite di CMake.

Per ora, sono disponibili i seguenti preset di configurazione:

- **windows-base**: Una configurazione Windows, che utilizza VS MSVC. Usa `lv_conf.h` come sistema di configurazione.
- **windows-kconfig**: Una configurazione Windows, che utilizza VS MSVC. Utilizza Kconfig come sistema di configurazione.
- **linux-base**: Una configurazione Linux, che utilizza Ninja e GCC. Usa `lv_conf.h` come sistema di configurazione.
- **linux-kconfig**: Una configurazione Linux, che utilizza Ninja e GCC. Utilizza Kconfig come sistema di configurazione.

E questi sono i preset della build:

- **windows-base\_dbg**: Build di Debug per Windows.
- **windows-base\_rel**: Build di Release per Windows.
- **linux-base\_dbg**: Build di Debug per Linux.
- **linux-base\_rel**: Build di Release per Linux.

Ecco come compilare utilizzando i preset:

```
cmake --preset windows-base
cmake --build --preset windows-base_dbg
ctest --preset windows-base_dbg
```

### Compilazione con IDE

Il metodo consigliato per utilizzare CMakePresets è un IDE compatibile con CMakePresets come

- VS 2022
- VS Code
- CLion

Basta caricare questo progetto nel proprio IDE e selezionare il preset desiderato.

### Compilazione con la GUI di CMake

Aprire questo progetto con l'interfaccia grafica di CMake e selezionare il preset desiderato. Premendo il pulsante *generate*, CMake creerà file di soluzione (per VS) o file Ninja (per la build Ninja per Linux).

Sono disponibili i seguenti target.

- lvgl (la libreria vera e propria, obbligatoria)
- lvgl\_thorvg (un'estensione per la grafica vettoriale, facoltativa)
- lvgl\_examples (esempi di utilizzo, facoltativi)
- lvgl\_demos (alcune demo, facoltative)

Tutti i target opzionali possono essere disabilitati impostando le variabili di cache appropriate. Se si utilizza cmake per installare lvgl, verranno create 3 cartelle.

- include/lvgl (contiene tutti gli header pubblici)
- bin (contiene tutti i binari (\*.dll))
- lib (contiene tutti i file sorgente precompilati (\*.lib))

### Integrare LVGL nel progetto usando cmake

Il sistema cmake di LVGL è progettato per essere integrato in progetti di livello superiore. Per farlo, è sufficiente aggiungerlo al file `CMakeLists.txt` del progetto.

Questo frammento aggiunge LVGL e necessita di un file `lv_conf.h` presente accanto alla cartella lvgl:

```
add_subdirectory(lvgl)
```

Questo frammento imposta LVGL e gli indica quale file `lv_conf.h` utilizzare:

```
set(LV_BUILD_CONF_PATH path/to/my_lv_conf.h)
add_subdirectory(lvgl)
```

Questo frammento imposta LVGL e punta alla cartella in cui si trova `lv_conf.h`:

```
set(LV_BUILD_CONF_DIR path/to/directory)
add_subdirectory(lvgl)
```

Questo frammento aggiunge LVGL e specifica di utilizzare Kconfig come sistema di configurazione:

```
set(LV_BUILD_USE_KCONFIG ON)
add_subdirectory(lvgl)
```

Questo frammento aggiunge LVGL e specifica di utilizzare Kconfig come sistema di configurazione e di utilizzare un defconfig specifico:

```
set(LV_BUILD_USE_KCONFIG ON)
set(LV_BUILD_DEFCONFIG_PATH path/to/my_defconfig)
add_subdirectory(lvgl)
```

Per abilitare le demo e gli esempi, impostare queste opzioni:

```
set(CONFIG_LV_BUILD_EXAMPLES ON)
set(CONFIG_LV_BUILD_DEMOS ON)
add_subdirectory(lvgl)
```

Di seguito è riportato un elenco delle opzioni/variabili disponibili

Variabile/Opzione	Tipo	Descrizione
LV_BUILD_CONF_PATH	PATH	Consente di impostare un percorso personalizzato per <code>lv_conf.h</code>
LV_BUILD_CONF_DIR	PATH	Consente di impostare una directory contenente <code>lv_conf.h</code>
LV_BUILD_USE_KCONFIG	BOOLEAN	Quando impostato, KConfig viene utilizzato come sorgente della configurazione. Questa opzione è disabilitata per default.
LV_BUILD_DEFCONFIG_PATH	PATH	Specificare di utilizzare un file .defconfig al posto del file .config corrente in una configurazione di Kconfig.
LV_BUILD_LVGL_H_SYSTEM_INCLUDE	BOOLEAN	Abilitare se LVGL verrà installato sul sistema o se il sistema di build utilizza una sysroot. L'attivazione di questa opzione implica che le risorse generate dallo script di generazione dell'immagine includeranno <code>lvgl.h</code> come inclusione di sistema. Ad esempio: <code>#include &lt;lvgl.h&gt;</code> . Questa opzione è disabilitata per default.
LV_BUILD_LVGL_H_SIMPLE_INCLUDE	BOOLEAN	Se abilitata, le risorse includeranno <code>lvgl.h</code> come semplice inclusione; questa opzione è abilitata di default.
LV_BUILD_SET_CONFIG_OPTS	BOOLEAN	Se abilitata, questa opzione esegue uno script che elabora la configurazione <code>lv_conf.h</code> /Kconfig utilizzando <code>pcpp</code> per generare le variabili CMake <code>CONFIG_LV_*</code> e <code>CONFIG_LV_BUILD_*</code> corrispondenti in base al contenuto di <code>lv_conf_internal.h</code> . Questo richiede <code>python3</code> con <code>venv</code> e <code>pip</code> o l'accesso a un <code>pcpp</code> funzionante. Se si utilizza KConfig, questa opzione viene abilitata automaticamente.
CONFIG_LV_BUILD_DEMOS	BOOLEAN	Se abilitato, compila le demo
CONFIG_LV_BUILD_EXAMPLES	BOOLEAN	Se abilitato, compila gli esempi
CONFIG_LV_USE_THORVG_INTERNAL	BOOLEAN	Se abilitato, viene compilata la versione LVGL in-tree di ThorVG
CONFIG_LV_USE_PRIVATE_API	BOOLEAN	Se abilitato, vengono installati sul sistema gli header privati <code>*_private.h</code>

#### Nota

Se `LV_BUILD_SET_CONFIG_OPTS` o `LV_BUILD_USE_KCONFIG` sono abilitati, le opzioni/variabili che iniziano con il prefisso `CONFIG_*` vengono automaticamente impostate sui valori presenti in `lv_conf.h`

## 4.4 Fornitori di chip

### 4.4.1 Alif

Alif Semiconductor è un fornitore di chip di nuova generazione che produce chip disponibili in una varietà di configurazioni. I loro chip sono progettati per eccellere sotto diversi aspetti contemporaneamente. Offrono combinazioni di prestazioni, basso consumo energetico, sicurezza e funzionalità speciali come l'intelligenza artificiale. Molti dei loro chip hanno due core asimmetrici. Un core è in genere ad alte prestazioni, mentre l'altro ad alta efficienza. Alif offre sia microcontrollori che microprocessori.

## LVGL su schede Alif

Questa è una guida per iniziare a usare LVGL su una scheda Alif. Descrive in dettaglio tutti i passaggi necessari per far funzionare il progetto di esempio LVGL `alif_m55-lvgl` su [Alif E7 Devkit Gen2](#); tuttavia, qualsiasi progetto basato sul [Alif VS Code Template](#) ha una procedura di installazione pressoché identica, quindi questa può essere utilizzata come guida generale per questi progetti. Esistono altri modi per compilare per le schede Alif, ad esempio con Zephyr RTOS. Vedere [Alif's GitHub repos](#).

Questa guida è per Linux e Windows.

Questo progetto utilizza l'accelerazione di rendering 2D D/AVE con D/AVE 2D [draw unit](#) di LVGL.

### Guida Passo-Passo

#### Installare Visual Studio Code

Installare Visual Studio code. Esistono diversi modi per installarlo a seconda della piattaforma. [Vedere qui](#).

##### Nota

I passaggi rimanenti possono essere eseguiti **facoltativamente** all'interno di un container Docker. È possibile connettersi al container Docker come container di sviluppo remoto VS Code.

Dovrebbero essere create due porte seriali al momento della connessione della scheda Alif. Su Linux, saranno simili a `/dev/ttyACM0` and `/dev/ttyACM1`. Nel comando `docker run` utilizzato per creare il container di sviluppo, includere `--device /dev/ttyACM0 --device /dev/ttyACM1` per consentire al container di accedere a quelle porte in modo da poter eseguire il flashing da esso.

Installare l'estensione VS Code "Dev Containers". Selezionare il container da "Remote Explorer" nel pannello laterale sinistro.

#### Installare gli strumenti Prerequisiti

Assicurarsi che siano installati nel proprio ambiente. Le estensioni VS Code si basano sulla loro presenza.

- `git`
- `curl`
- `unzip`

#### Installare gli Alif SE Tool

Creare un account Alif e scaricare gli strumenti da [here](#) sotto "Alif Security Toolkit". Estrarre il file. Il path in cui è stato estratto sarà necessario in seguito.

Su Linux, l'estrazione può essere eseguita eseguendo quanto segue:

```
cd
tar -xf Downloads/APFW0002-app-release-exec-linux-SE_FW_1.101.00_DEV.tar
ls
pwd
```

Tra i risultati di `ls` si dovrebbe vedere `app-release-exec-linux`. Questo, combinato con l'output di `pwd`, è il percorso che si dovrà utilizzare in seguito. Ovvero, `/home/you/app-release-exec-linux`.

#### Installare il Software J-Link (facoltativo)

Scaricare l'ultima versione stabile del software [J-Link Software](#). Il path di installazione sarà necessario in seguito.

## Clonare il Progetto alif\_m55-lvgl

```
git clone --recursive https://github.com/alifsemi/alif_m55-lvgl
```

## Aprire alif\_m55-lvgl in VS Code

Aprire il repository clonato in VS Code. Affinché le estensioni di VS Code funzionino correttamente, si consiglia di aprire la cartella in VS Code anziché aprire una directory padre contenente il progetto.

```
code alif_m55-lvgl
```

oppure si va su **File > Open Folder** in VS Code e si apre **alif\_m55-lvgl**.

Se viene richiesto di installare automaticamente le estensioni consigliate, cliccare su "Install" per saltare il passaggio successivo.

## Installare le Estensioni di VS Code Richieste

Installare le seguenti estensioni di VS Code dal pannello laterale "Extensions"

- Arm Tools Environment Manager
- Arm CMSIS Solution
- C/C++ Extension Pack
- Cortex-Debug (facoltativo necessario per il debug)

## Activate Environment

Se non è avvenuto automaticamente, cliccare su "Arm Tools" nella barra in basso e poi su "Activate Environment" nell'elenco che appare. Verranno installati CMake, ninja-build, un compilatore GCC ARM e cmsis-toolbox.

Se si vede solo "Reactivate Environment", è probabile che sia già attivo.

## Impostare i Path dei Tool Installati

Premere ctrl + shift + p. Digitare "preferences" e selezionare l'opzione "Preferences: Open User Settings (JSON)" tra le opzioni. Si aprirà un file **settings.json**. Nota: se si utilizza un container Docker, è meglio modificare il file JSON nella directory del progetto in **.vscode/settings.json**.

Si devono aggiungere alcune voci (almeno "**alif.setools.root**") al JSON che si vede.

Se il **settings.json** inizialmente appare così...

```
{
  "workbench.colorTheme": "Default Dark+",
  "editor.renderWhitespace": "all",
}
```

... allora dovrebbe apparire così:

```
{
  "workbench.colorTheme": "Default Dark+",
  "editor.renderWhitespace": "all",
  "alif.setools.root": "C:/alif-se-tools/app-release-exec",
  "cortex-debug.JLinkGDBServerPath": "C:/Program Files/SEGGER/JLink/JLinkGDBServerCL.exe"
}
```

Quanto sopra usa i path Windows come esempio. Un path Linux per i Tool Alif SE potrebbe essere simile a **/home/you/app-release-exec-linux**.

## Configurare la Variante della Board

Aprire il file **board.h**.

Identificare la variante della scheda nell'elenco e impostare **BOARD\_ALIF\_DEVKIT\_VARIANT** sul valore corretto. Potrebbe essere necessario impostare anche **BOARD\_ILI9806E\_PANEL\_VARIANT** se il valore predefinito non corrisponde al proprio.

## Impostare il Build Context, Compilare e Flash-are

Accedere alla vista "Manage Solution" dall'estensione CMSIS Solution. Vi si può accedere cliccando sull'icona a forma di ingranaggio nella barra inferiore oppure andando al pannello CMSIS a sinistra e cliccando sull'ingranaggio in alto. Una volta aperto, si presenta come una scheda dell'editor grafico chiamata "Manage Solution" con un'icona a forma di ingranaggio.

### Importante

In "Run and Debug" > "Run Configuration" (column) > "alif" (row), cliccare sul menù a discesa e selezionare "First time pack installation". Cliccare sull'icona di riproduzione nella parte superiore del pannello laterale sinistro di CMSIS. Nel terminale si dovrebbero vedere i pacchetti CMSIS in fase di installazione. Attendere il completamento.

Ora si può cliccare sull'icona del martello accanto all'icona di "play" per compilare il progetto. Verranno compilate alcune centinaia di file. Attendere il completamento.

Aprire il menù a discesa precedente e selezionare "Prepare program with Security Toolkit". Cliccare sull'icona di "play" [riproduzione]. Vengono preparati alcuni file come passaggio preliminare al flashing.

Aprire di nuovo il menu a discesa e selezionare "Program with Security Toolkit". Cliccare sull'icona di "play" [riproduzione]. Se è la prima volta, verrà chiesto di scegliere quale porta seriale utilizzare per flashare la scheda. Si può provare `/dev/ttyACM0`. Se è quella sbagliata, l'operazione fallirà e si dovrà aprire il menù a discesa e selezionare "Program with Security Toolkit (select COM port)" per sovrascrivere quella precedente che è stata salvata come default.

Il benchmark LVGL dovrebbe essere eseguito sulla scheda Alif al termine del flashing.

## Ulteriori Informazioni

In caso di difficoltà durante la lettura di questa guida, fare riferimento a queste fonti Alif per passaggi più dettagliati.

- [Getting Started with VSCode CMSIS pack project](#)
- [VSCode Getting Started Template](#)

È possibile scaricare la "Alif Security Toolkit Quick Start Guide" da <https://alifsemi.com/support/software-tools/ensemble/>, presupponendo che sia stato creato un account, per imparare a utilizzare gli strumenti Alif SE per eseguire manipolazioni di basso livello sulla scheda.

## I Core HP e HE e Build Ottimizzata

Nella vista "Manage Solution" illustrata nella guida, è possibile selezionare un target HP o un target HE. Questi si riferiscono ai due core distinti presenti nell'Alif E7. "HE" sta per "High Efficiency" mentre "HP" sta per "High Performance". Per ottenere le migliori prestazioni da un'applicazione LVGL, selezionare HP. Considerare HE quando il consumo energetico è un problema. Il vantaggio di avere core asimmetrici è che l'applicazione può teoricamente eseguire carichi di lavoro a bassa priorità in modo efficiente sul core HE e delegare carichi di lavoro critici e ad alta intensità di elaborazione al core HP.

È anche possibile scegliere un "Build Type". Per prestazioni ottimali, scegliere "release". Se si esegue il debug, scegliere "debug".

Per massimizzare il punteggio nel benchmark LVGL e massimizzare le prestazioni di un'applicazione LVGL in generale, deve essere selezionato il core HP e il tipo di build dev'essere release.

### 4.4.2 Arm

Arm è un'azienda leader nella progettazione di semiconduttori e software, rinomata per aver creato i core dei microcontrollori (MCU) Cortex-M e quelli dei processori Cortex-A/R (MPU), che sono parte integrante di un'ampia gamma di dispositivi. Questi core sono il cuore di molti sistemi embedded e alimentano chip di giganti del settore come STMicroelectronics, NXP e Renesas. I progetti a basso consumo energetico di Arm sono utilizzati in miliardi di dispositivi in tutto il mondo, dai microcontrollori agli smartphone e ai server. Concedendo in licenza i propri progetti di processori, Arm consente a un ampio ecosistema di partner di sviluppare soluzioni personalizzate ottimizzate per prestazioni, consumo energetico e dimensioni. L'architettura di Arm è altamente compatibile con vari sistemi

operativi e librerie software, tra cui LVGL, rendendola una scelta versatile per gli sviluppatori che creano interfacce utente grafiche efficienti e ad alte prestazioni.

### Compilare LVGL per Arm

Non è richiesta alcuna azione specifica. Qualsiasi compilatore che supporti l'architettura Arm come target può essere utilizzato per compilare il codice sorgente di LVGL, inclusi GCC, LLVM e AC6.

È anche possibile cross-compile LVGL per una MPU (invece di compilarlo sull'hardware target) o creare una libreria shared. Per ulteriori informazioni, consultare [CMake](#).

### Introduzione a AC6

Poiché AC6 è una toolchain proprietaria, contiene molte ottimizzazioni specifiche, quindi è possibile aspettarsi le migliori prestazioni quando lo si utilizza.

AC6 non è gratuito, ma offre una licenza community che può essere attivata come segue:

1. Scaricare e installare il compilatore AC6 dal [sito web di Arm](#).
2. Per registrare una licenza community, andare alla cartella `bin` del compilatore e, in un terminale, eseguire `armlm.exe activate -server https://mdk-preview.keil.arm.com -product KEMDK-COM0` (Su Linux, usare `./armlm`).

### Supporto IDE

Non ci sono limitazioni sugli IDE supportati. LVGL funziona negli IDE di vari fornitori, tra cui Keil MDK di Arm, IAR, e2 studio di Renesas, MCUXpresso di NXP, CubeIDE di ST, nonché progetti custom make o CMake.

### Arm2D e il set di istruzioni Helium

Arm Cortex-M55 e Cortex-M85 dispongono del set di istruzioni SIMD Helium. Tra le altre cose, questo può effettivamente velocizzare il rendering dell'interfaccia utente. [Arm2D](#) è una libreria gestita da Arm che sfrutta il set di istruzioni Helium.

Notare che GCC presenta alcuni problemi noti con le "Helium intrinsics". Si consiglia di utilizzare AC6 o LLVM quando si utilizza codice Helium.

Per aggiungere Arm2D al progetto, seguire questi passaggi:

1. Per sfruttarne la potenza, assicurarsi che `mcpu` sia impostato su `cortex-m85`, `cortex-m55`, o `cortex-m52` e aggiungere il flag `-fvectorize`. Per testare senza SIMD, si usa ad esempio `cortex-m85+nomve`.
2. Arm2D può essere scaricato da <https://github.com/ARM-software/Arm-2D>. Valutare la possibilità di utilizzare il branch `developing`, che contiene gli aggiornamenti più recenti.
3. Aggiungere `Arm-2D/Library/Include` ai path degli "include".
4. Copiare `Arm-2D/Library/Include/template/arm_2d_cfg.h` nella posizione che si preferisce per fornire la configurazione di default per Arm2D. La cartella che contiene `arm_2d_cfg.h` dev'essere aggiunta al path dell'include.
5. Il repository Arm2D contiene diversi esempi e template; tuttavia, assicurarsi che venga compilato solo `Arm-2D/Library/Source`.
6. Anche la libreria CMSIS DSP deve essere aggiunta al progetto. Si può utilizzare CMSIS-PACKS o aggiungerla manualmente.
7. Per prestazioni migliori, abilitare LT0 (Link Time Optimization) e usare `-Omax` o `-Ofast`.
8. Arm2D tenta di leggere/scrivere più dati con una singola istruzione. Pertanto, è importante utilizzare la memoria più veloce (ad esempio, BSS o TCM) per il buffer di LVGL per evitare colli di bottiglia nella larghezza di banda della memoria.

9. Abilitare `LV_USE_DRAW_ARM2D_SYNC 1` e `LV_USE_DRAW_SW_ASM LV_DRAW_SW_ASM_HE-LIUM` in `lv_conf.h`.

## Accelerazione Neon

Diversi microprocessori Cortex-A supportano il set di istruzioni Neon SIMD. LVGL ha un supporto integrato per migliorare le prestazioni del rendering software utilizzando le istruzioni Neon. Per abilitare l'accelerazione Neon, impostare `LV_USE_DRAW_SW_ASM` to `LV_DRAW_SW_ASM_NEON` in `lv_conf.h`.

### 4.4.3 Espressif (Serie di Chip ESP32)

LVGL può essere utilizzato e configurato come componente standard [ESP-IDF](#).

Se non si ha familiarità con ESP-IDF, seguire le istruzioni nella [ESP-IDF Programming guide](#) per installare e configurare ESP-IDF sul computer.

## Progetti demo LVGL per ESP32

Per un rapido avvio con LVGL ed ESP32, sono disponibili i seguenti progetti demo preconfigurati per schede di sviluppo specifiche:

- [ESP-BOX-3](#)
- [ESP32-S3-LCD-EV-BOARD](#)
- [M5Stack-CoreS3](#)

Fare riferimento ai file README.md in questi repository per le istruzioni di compilazione e flashing.

Questi progetti demo utilizzano i Board Support Package (BSP) di Espressif. Ulteriori BSP ed esempi sono disponibili nel repository [esp-bsp](#).

## Utilizzo di LVGL nel proprio progetto ESP-IDF

Il modo più semplice per integrare LVGL nel proprio progetto ESP-IDF è tramite il componente [esp\\_lvgl\\_port](#). Questo componente, utilizzato nei progetti demo menzionati sopra, fornisce funzioni di supporto per una facile installazione di LVGL e driver di display. Inoltre, può aggiungere il supporto per touch, encoder rotativi, pulsanti o ingressi USB HID. Semplifica il risparmio energetico, la rotazione dello schermo e altre funzionalità specifiche della piattaforma.

`esp_lvgl_port` supporta le versioni LVGL 8 e 9 ed è compatibile con ESP-IDF v4.4 e successive. Per aggiungerlo al progetto, si usa il seguente comando:

```
idf.py add-dependency "espressif/esp_lvgl_port^2.3.0"
```

Per default, `esp_lvgl_port` dipende dall'ultima versione stabile di LVGL, quindi non sono necessari passaggi aggiuntivi per i nuovi progetti. Se è richiesta una versione specifica di LVGL, specificatela nel progetto per evitare aggiornamenti automatici. LVGL può essere utilizzato anche senza `esp_lvgl_port`, come descritto di seguito.

## Ottenere LVGL

LVGL è distribuito tramite [ESP Registry](#), dove vengono caricate tutte le release LVGL. Se non si desidera utilizzare `esp_lvgl_port`, è possibile aggiungere [LVGL component](#) al progetto con il seguente comando:

```
idf.py add-dependency "lvgl/lvgl^9.*"
```

Modificare la parte `^9.*` in base ai requisiti di versione LVGL. Ulteriori informazioni sulle specifiche di versione sono disponibili nella documentazione di [IDF Component Manager](#). Durante la build successiva, il componente LVGL verrà recuperato dal registro dei componenti e aggiunto al progetto.

### Uso avanzato: Utilizzare LVGL come componente locale

Per lo sviluppo e il test di LVGL, potrebbe essere utile utilizzare LVGL come componente locale anziché dal Registro ESP, che offre solo versioni rilasciate e non consente modifiche locali. Per farlo, clonare LVGL nel proprio progetto con il seguente comando:

```
git submodule add https://github.com/lvgl/lvgl.git components/lvgl
```

### Nota

Tutti i componenti da \${project\_dir}/components vengono aggiunti automaticamente alla build.

## Configurazione

Per configurare LVGL, avviare il menù di configurazione con `idf.py menuconfig` nella directory principale del progetto. Accedi a `Component config` e poi a `LVGL configuration`.

## Supporto per Driver di Display e Touch

Per un progetto LVGL di successo, ci sarà bisogno di un driver display e, facoltativamente, di un driver touch. Espressif fornisce questi driver basati sul suo componente `esp_lcd`.

- `esp_lcd` supporta nativamente alcuni `display` di base
- Altri display sono mantenuti nel repository `esp-bsp repository` e vengono caricati nell'ESP Registry
- I driver touch sono mantenuti nel repository `esp-bsp repository` e vengono caricati nell'ESP Registry

Questi componenti condividono un'API pubblica comune, semplificando la migrazione dei progetti tra diversi driver di display e touch.

Per aggiungere un driver di display o touch al progetto, utilizzare un comando come:

```
idf.py add-dependency "espressif/esp_lcd_gc9a01^2.0.0"
```

## Utilizzo del file system in ESP-IDF

ESP-IDF utilizza le funzioni standard C per le operazioni sui file (`fopen`, `fread`) in tutte le sue API relative all'archiviazione. Ciò consente un'interoperabilità perfetta con LVGL quando si abilita la configurazione `LV_USE_FS_STDIO`. Il processo è descritto in dettaglio di seguito, utilizzando SPIFFS come dimostrazione.

### • Decidere quale sistema di archiviazione utilizzare

ESP-IDF offre molti esempi pronti all'uso, come `SPIFFS`, `SD Card` e `LittleFS`.

### • Riconfigurare il progetto

Il progetto di esempio dovrebbe essere esaminato per i dettagli, ma in generale le modifiche riguardano:

- Abilitazione del file system STDIO di LVGL nella configurazione

È possibile utilizzare `menuconfig`:

- \* `Component config` → `LVGL configuration` → `3rd Party Libraries`: enable `File system on top of stdio API`
- \* Poi selezionare `Set an upper cased letter on which the drive will accessible` e impostarla su `65` (ASCII A)
- \* È anche possibile impostare `Default driver letter` a `65` per ignorare il prefisso nei path dei file.

- Modifica della tabella delle partizioni

La configurazione esatta dipende dalle dimensioni della flash e dalle partizioni esistenti, ma il nuovo risultato finale dovrebbe essere simile a questo:

Tabella 1: Partition Table

nvs	data	nvs	0x9000	0x6000
phy_init	data	phy	0xf000	0x1000
factory	app	factory	0x10000	1400k
storage	data	spiffs		400k

**Nota**

Se non si utilizza ancora un file `partition.csv` personalizzato, è possibile aggiungerlo tramite menuconfig (Partition Table → Partition Table → Custom partition table CSV).

- Applicare le modifiche al sistema di build

Alcuni file system ESP forniscono la generazione automatica da una cartella host tramite CMake. Le righe corrette devono essere copiate in `main/CMakeLists.txt`

**Nota**

`LittleFS` ha dipendenze aggiuntive che devono essere aggiunte a `main/idf_component.yml`

- **Preparare i file delle immagini**

Il tool Python `LVGLImage.py` di LVGL può essere utilizzato per convertire le immagini in file binari di pixel map. Supporta vari formati e compressioni.

Mentre librerie di terze parti (come `LodePNG` e `Tiny JPEG`) consentono di utilizzare file immagine senza conversione.

Dopo aver preparato i file, è necessario spostarli sul dispositivo di destinazione:

- Se attivato correttamente, un file system **SPIFFS** basato sulla cartella `spiffs_image` dovrebbe essere generato automaticamente e successivamente flashato sul dispositivo di destinazione.
- Un meccanismo simile per **LittleFS** usa la cartella `flash_data`, ma è disponibile solo per host Linux.
- Per la **SD Card**, è possibile utilizzare un file browser tradizionale.

- *Invoca le chiamate API appropriate nel codice dell'applicazione\**

Le funzionalità principali richiedono solo poche righe. L'esempio seguente disegna anche l'immagine.

```
#include "esp_spiffs.h"

void lv_example_image_from_esp_fs(void) {
    esp_vfs_spiffs_conf_t conf = {
        .base_path = "/spiffs",
        .partition_label = NULL,
        .max_files = 5,
        .format_if_mount_failed = false
    };
    esp_err_t ret = esp_vfs_spiffs_register(&conf);

    if (ret != ESP_OK) {
        ESP_LOGE(TAG, "Failed to register SPIFF filesystem");
        return;
    }

    lv_obj_t * obj = lv_image_create(lv_screen_active());
    lv_image_set_src(widget, "A:/spiffs/logo.bin");
    lv_obj_center(widget);
}
```

- **Build e flash**

Dopo aver chiamato `idf.py build flash`, l'immagine dovrebbe essere visualizzata sullo schermo.

### Nota

Le modifiche apportate da `menuconfig` non vengono tracciate nel repository se il file  `sdkconfig` viene aggiunto a `.gitignore`, che è l'impostazione di default per molti progetti ESP-IDF. Per rendere permanente la configurazione, aggiungere le seguenti righe a  `sdkconfig.defaults`:

```
CONFIG_PARTITION_TABLE_CUSTOM=y
CONFIG_LV_USE_FS_STDIO=y
CONFIG_LV_FS_STDIO LETTER=65
CONFIG_LV_FS_DEFAULT_DRIVER LETTER=65
```

## Supporto per Pixel Processing Accelerator

Alcune serie di chip ESP32, come l'ESP32-P4, supportano l'hardware Pixel Processing Accelerator (PPA), in grado di accelerare le operazioni di riempimento e fusione delle immagini. Questa periferica funziona con l'hardware DMA-2D, responsabile dello spostamento dei buffer di input/output da/verso il motore di elaborazione PPA.

### Dispositivi supportati

I target Espressif che supportano il PPA sono:

- Le serie ESP32-P4.

### Utilizzo del PPA nel progetto ESP-IDF

LVGL supporta, a livello sperimentale, l'accelerazione del riempimento e della fusione delle immagini tramite PPA. L'utente può abilitarla all'interno del proprio  `sdkconfig.default` aggiungendo la seguente opzione per abilitare l'unità di disegno PPA insieme al rendering software:

```
CONFIG_LV_USE_PPA=y
```

Salvare il file e quindi ricostruire il progetto. Sarà sufficiente aggiungere il codice PPA e l'esecuzione inizierà automaticamente, quindi non sono necessari ulteriori passaggi dal punto di vista del codice utente.

### Limitazioni

Si noti che il PPA è a livello sperimentale, dove sono previsti alcuni miglioramenti delle prestazioni nelle attività di disegno relative alla copia o al riempimento di rettangoli, mentre per la fusione delle immagini, sebbene operativa, non si registrano miglioramenti significativi; la causa iniziale, secondo la sezione PPA del manuale di riferimento, è dovuta alla larghezza di banda della memoria DMA-2D.

### 4.4.4 NXP

NXP ha integrato LVGL nei pacchetti "SDK MCUXpresso" per diversi dei nostri microcontrollori come componente software opzionale, consentendo una facile valutazione e migrazione nella progettazione del prodotto. LVGL è una libreria grafica embedded gratuita e open source con funzionalità che consentono di creare GUI embedded con elementi grafici intuitivi, splendidi effetti visivi e un ridotto utilizzo di memoria. Il framework grafico completo include una varietà di widget da utilizzare nella creazione della GUI e supporta funzioni più avanzate come animazioni e anti-aliasing.

LVGL abilita la grafica nel nostro tool gratuito "GUI Guider". È disponibile per l'uso con i microcontrollori generici e crossover di NXP, fornendo agli sviluppatori uno strumento per creare applicazioni GUI complete e di alta qualità con LVGL.

## Creazione di un nuovo progetto con LVGL

Scaricare subito un [SDK per una board supportata](#) e iniziare a sviluppare la prossima applicazione GUI. È completamente configurato con LVGL (e con supporto PXP/VGLite/G2D se i moduli sono presenti), non è richiesto alcun ulteriore lavoro di integrazione.

## Accelerazione HW per piattaforme NXP iMX RT

A seconda della piattaforma RT utilizzata, l'accelerazione può essere eseguita da NXP PXP (PiXel Pipeline) e/o dalla GPU Verisilicon tramite un'API denominata VGLite. Ogni acceleratore ha un proprio contesto che ne consente l'utilizzo sia singolarmente che simultaneamente (in modalità multithreading LVGL).

## Accelerazione HW per piattaforme NXP iMX

Sulle piattaforme MPU, l'accelerazione può essere eseguita (indipendentemente dall'hardware) dalla libreria NXP G2D. Questo acceleratore ha un proprio contesto che ne consente l'utilizzo sia singolarmente che simultaneamente con la CPU (in modalità multithreading LVGL).

## Acceleratore PXP

### Configurazione di base:

- Selezionare il motore PXP NXP in "lv\_conf.h": Impostare `LV_USE_PXP` a 1.
- Per utilizzare PXP come unità di disegno, selezionare in "lv\_conf.h": Impostare `LV_USE_DRAW_PXP` a 1.
- Per utilizzare PXP per ruotare lo schermo, selezionare in "lv\_conf.h": Impostare `LV_USE_ROTATE_PXP` a 1.
- Enable PXP asserts in "lv\_conf.h": Set :c:macro: `LV_USE_PXP_ASSERT` to 1. There are few PXP assertions that can stop the program execution in case the `LV_ASSERT_HANDLER` is set to `while(1);` (Halt by default). Else, there will be logged just an error message via `LV_LOG_ERROR`.
- Se è definito il simbolo `SDK_OS_FREE_RTOS`, verrà utilizzata l'implementazione di FreeRTOS, altrimenti verrà incluso codice bare metal.

### Inizializzazione di base:

L'inizializzazione del disegno PXP viene eseguita automaticamente in `lv_init()` once the PXP is enabled as a draw unit or to rotate the screen, no user code is required:

```
#if LV_USE_DRAW_PXP || LV_USE_ROTATE_PXP
    lv_draw_pxp_init();
#endif
```

Durante l'inizializzazione del PXP, verrà creata una nuova unità di disegno `lv_draw_pxp_unit_t` con le callback aggiuntive, se `LV_USE_DRAW_PXP` è impostato a 1:

```
lv_draw_pxp_unit_t * draw_pxp_unit = lv_draw_create_unit(sizeof(lv_draw_pxp_unit_t));
draw_pxp_unit->base_unit.evaluate_cb = _pxp_evaluate;
draw_pxp_unit->base_unit.dispatch_cb = _pxp_dispatch;
draw_pxp_unit->base_unit.delete_cb = _pxp_delete;
```

e verrà generato un thread aggiuntivo `_pxp_render_thread_cb()` per gestire i task di disegno supportati.

```
#if LV_USE_PXP_DRAW_THREAD
    lv_thread_init(&draw_pxp_unit->thread, "pxpdraw", LV_THREAD_PRIO_HIGH, _pxp_render_thread_cb, 2 * 1024, draw_pxp_unit);
#endif
```

Se `LV_USE_PXP_DRAW_THREAD` non è definito, non verrà creato alcun thread di disegno aggiuntivo e il task di disegno PXP verrà eseguito sullo stesso thread principale LVGL.

`_pxp_evaluate()` verrà chiamato dopo la creazione di ogni attività e analizzerà se l'attività è supportata da PXP o meno. Se supportato, all'attività verranno assegnati un punteggio preferito e l'ID dell'unità di disegno. Uno `score` uguale a 100 è lo score di default della CPU. Uno score più piccolo indica che PXP è in grado di disegnarlo più velocemente.

`_pxp_dispatch()` è la callback del dispatcher PXP, che accetta un task pronto per il disegno (con `DRAW_UNIT_ID_PXP` impostato) e la passa all'unità di disegno PXP per l'elaborazione.

`_pxp_delete()` pulisce l'unità di disegno PXP.

### Funzionalità supportate:

Diverse funzionalità di disegno in LVGL possono essere demandate al motore PXP. La CPU è disponibile per altre operazioni mentre PXP è in esecuzione. È necessario un RTOS per bloccare il thread di disegno LVGL e passare a un altro task o sospendere la CPU per risparmiare energia.

Le attività di disegno supportate sono disponibili in "src/draw/nxp/pxp/lv\_draw\_pxp.c":

```
switch(t->type) {
    case LV_DRAW_TASK_TYPE_FILL:
        lv_draw_pxp_fill(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_IMAGE:
        lv_draw_pxp_img(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_LAYER:
        lv_draw_pxp_layer(t, t->draw_dsc, &t->area);
        break;
    default:
        break;
}
```

Inoltre, la rotazione dello schermo può essere gestita da PXP:

- Riempire l'area con un colore (senza raggio, senza gradiente) + opacità opzionale.
- Blit [Block Image Transfer] dell'immagine sorgente RGB565/ARGB888/XRGB8888 sulla destinazione. RGB565/RGB888/ARGB888/XRGB8888 + opacità opzionale.
- Ricolorare l'immagine sorgente RGB565.
- Ridimensionare e ruotare (90, 180, 270 gradi) l'immagine sorgente RGB565.
- Fusione dei livelli (con gli stessi formati supportati per il blitting).
- Ruotare lo schermo (90, 180, 270 gradi).

### Limitazioni note:

- PXP può ruotare i fotogrammi solo di angoli multipli di 90 gradi.
- La rotazione non è supportata per le immagini non allineate a blocchi di 16x16 pixel. PXP è impostato per elaborare blocchi 16x16 per ottimizzare il sistema in termini di larghezza di banda di memoria e tempo di elaborazione delle immagini. Il motore di output essenzialmente tronca tutti i pixel di output dopo che è stato scritto il numero desiderato di pixel. Quando si ruota un'immagine sorgente e l'output non è divisibile per la dimensione del blocco, i pixel errati potrebbero essere troncati e l'immagine di output finale potrebbe apparire spostata.
- La ricolorazione o la trasformazione di immagini con opacità o canale alfa non possono essere ottenute in una singola configurazione della pipeline PXP. Sarebbero necessari due o più passaggi.
- L'indirizzo del buffer deve essere allineato a 64 byte: impostare `LV_DRAW_BUF_ALIGN` a 64 in "lv\_conf.h". Non è richiesto alcun allineamento di passo: impostare `LV_DRAW_BUF_STRIDE_ALIGN` a 1 in "lv\_conf.h".

### Setup del progetto:

- Aggiungere i file sorgente relativi a PXP (e i relativi header, se disponibili) al progetto:
  - "src/draw/nxp/pxp/lv\_draw\_buf\_pxp.c": callback del buffer di disegno
  - "src/draw/nxp/pxp/lv\_draw\_pxp\_fill.c": area di riempimento
  - "src/draw/nxp/pxp/lv\_draw\_pxp\_img.c": immagine blit (con ricolorazione o trasformazione opzionale)
  - "src/draw/nxp/pxp/lv\_draw\_pxp\_layer.c": fusione dei livelli
  - "src/draw/nxp/pxp/lv\_draw\_pxp.c": inizializzazione dell'unità di disegno

- "src/draw/nxp/pxp/lv\_pxp\_cfg.c": init,\_deinit, run/wait del dispositivo PXP
- "src/draw/nxp/pxp/lv\_pxp\_osa.c": astrazione del sistema operativo (FreeRTOS o bare metal)
- "src/draw/nxp/pxp/lv\_pxp\_utils.c": helper delle funzioni
- Il codice relativo a PXP dipende da due driver forniti dall'SDK MCU. Questi driver devono essere aggiunti al progetto:
  - fsl\_pxp.c: Driver PXP
  - fsl\_cache.c: Funzioni di gestione della cache della CPU

### Configurazione di default di PXP:

- L'implementazione dipende da più funzioni specifiche del sistema operativo. La struttura `pxp_cfg_t` con puntatori di callback viene utilizzata come parametro per la funzione `lv_pxp_init()`. Implementazione di default per FreeRTOS in `lv_pxp_osa.c`.
  - `pxp_interrupt_init()`: Inizializza l'interrupt PXP (configurazione HW, configurazione del sistema operativo)
  - `pxp_interrupt_deinit()`: Deinizializza l'interrupt PXP (configurazione HW, configurazione del sistema operativo)
  - `pxp_run()`: Avvia il processo PXP. Utilizza un meccanismo specifico del sistema operativo per bloccare il thread di disegno.
  - `pxp_wait()`: Attende il completamento di PXP.

### Acceleratore VGLite

Il motore VGLite può gestire funzionalità di disegno aggiuntive in LVGL. La CPU è disponibile per altre operazioni mentre VGLite è in esecuzione. È necessario un RTOS per bloccare il thread di disegno LVGL e passare a un altro task o sospendere la CPU per risparmiare energia.

### Configurazione di base:

- Selezionare il motore NXP VGLite in "lv\_conf.h": Impostare `LV_USE_DRAW_VGLITE` a 1. Il simbolo `SDK_OS_FREE_RTOS` deve essere definito in modo che l'implementazione "osal" [OS Abstraction Layer] del driver FreeRTOS sia abilitata.
- Enable VGLite asserts in "lv\_conf.h": Set :c:macro: `LV_USE_VGLITE_ASSERT` to 1. VGLite assertions will verify the driver API status code and in any error, it can stop the program execution in case the :c:macro: `LV_ASSERT_HANDLER` is set to `while(1);` (Halt by default). Else, there will be logged just an error message via `LV_LOG_ERROR`.

### Inizializzazione di base:

Inizializza la GPU VGLite prima di chiamare `lv_init()` specificando la larghezza/altezza della finestra di tessellatura. I valori di default per la larghezza e l'altezza della tessellatura e la dimensione del buffer dei comandi si trovano nel file SDK "vglite\_support.h".

```
#if LV_USE_DRAW_VGLITE
#include "vg_lite.h"
#include "vglite_support.h"
#endif

#if LV_USE_DRAW_VGLITE
if(vg_lite_init(DEFAULT_VG_LITE_TW_WIDTH, DEFAULT_VG_LITE_TW_HEIGHT) != VG_LITE_SUCCESS)
{
    PRINTF("VGLite init error. STOP.");
    vg_lite_close();
    while (1)
    ;
}

if (vg_lite_set_command_buffer_size(VG_LITE_COMMAND_BUFFER_SIZE) != VG_LITE_SUCCESS)
{
    PRINTF("VGLite set command buffer. STOP.");
    vg_lite_close();
    while (1)
}

```

(continues on next page)

(continua dalla pagina precedente)

```
};  
#endif
```

L'inizializzazione del disegno di VGLite viene eseguita automaticamente in `lv_init()` una volta abilitato VGLite, non è richiesto alcun codice utente:

```
#if LV_USE_DRAW_VGLITE  
    lv_draw_vglite_init();  
#endif
```

Durante l'inizializzazione di VGLite, verrà creata una nuova unità di disegno `lv_draw_vglite_unit_t` con le callback aggiuntive:

```
lv_draw_vglite_unit_t * draw_vglite_unit = lv_draw_create_unit(sizeof(lv_draw_vglite_unit_t));  
draw_vglite_unit->base_unit.evaluate_cb = _vglite_evaluate;  
draw_vglite_unit->base_unit.dispatch_cb = _vglite_dispatch;  
draw_vglite_unit->base_unit.delete_cb = _vglite_delete;
```

e verrà generato un thread aggiuntivo `_vglite_render_thread_cb()` per gestire le attività di disegno supportate.

```
#if LV_USE_VGLITE_DRAW_THREAD  
    lv_thread_init(&draw_vglite_unit->thread, "vglitedraw", LV_THREAD_PRIO_HIGH, _vglite_render_thread_cb, 2 * 1024, draw_vglite_unit);  
#endif
```

Se `LV_USE_VGLITE_DRAW_THREAD` non è definito, non verrà creato alcun thread di disegno aggiuntivo e l'attività di disegno di VGLite verrà eseguita sullo stesso thread principale di LVGL.

`_vglite_evaluate()` verrà chiamato dopo la creazione di ogni attività e analizzerà se l'attività è supportata da VGLite o meno. Se supportato, all'attività verranno assegnati un punteggio preferito e l'ID dell'unità di disegno. Uno `score` uguale a `100` è lo score di default della CPU. Uno score più piccolo indica che VGLite è in grado di disegnarlo più velocemente.

`_vglite_dispatch()` è la callback del dispatcher VGLite, che accetta un task pronto per il disegno (con `DRAW_UNIT_ID_VGLITE` impostato) e la passa all'unità di disegno VGLite per l'elaborazione.

`_vglite_delete()` pulirà l'unità di disegno VGLite.

### Configurazione avanzata:

- Abilitare la divisione blit VGLite in "lv\_conf.h": Impostare :c:macro: `LV_USE_VGLITE_BLIT_SPLIT` a 1. Abilitando la soluzione alternativa per la divisione blit si attenuerà qualsiasi problema di degradazione della qualità su schermi con dimensioni superiori a 352 pixel.

```
#define VGLITE_BLIT_SPLIT_THR 352
```

- Per default, la soglia del "blit split" è impostata su 352. I blit [Block Image Transfer] con larghezza o altezza superiori a questo valore verranno eseguiti in più passaggi. Il valore deve essere un multiplo dell'allineamento "stride" in px. Per la maggior parte dei formati colore, l'allineamento è di 16 px (ad eccezione dei formati indice). La trasformazione non sarà supportata una volta con la divisione blit.
- Abilitare l'attività di disegno VGLite in modo sincrono in "lv\_conf.h": Impostare :c:macro: `LV_USE_VGLITE_DRAW_ASYNC` a 1. È possibile mettere in coda più task di disegno e scaricarle una volta sulla GPU in base allo stato di inattività della GPU. Se la GPU è occupata, l'attività verrà messa in coda e il dispatcher VGLite richiederà una nuova attività disponibile. Se la GPU è inattiva, la coda con eventuali attività in sospeso verrà scaricata sulla GPU. Lo stato di completamento dell'attività di disegno verrà inviato al thread LVGL principale in modo asincrono.

### Funzionalità supportate:

Diverse funzionalità di disegno in LVGL possono essere scaricate sul motore VGLite. La CPU è disponibile per altre operazioni mentre la GPU è in esecuzione. È necessario un RTOS per bloccare il thread di disegno LVGL e passare a un'altra attività o sospendere la CPU per risparmiare energia.

Le attività di disegno supportate sono disponibili in "src/draw/nxp/vglite/lv\_draw\_vglite.c":

```

switch(t->type) {
    case LV_DRAW_TASK_TYPE_LABEL:
        lv_draw_vglite_label(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_FILL:
        lv_draw_vglite_fill(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_BORDER:
        lv_draw_vglite_border(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_IMAGE:
        lv_draw_vglite_img(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_ARC:
        lv_draw_vglite_arc(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_LINE:
        lv_draw_vglite_line(t, t->draw_dsc);
        break;
    case LV_DRAW_TASK_TYPE_LAYER:
        lv_draw_vglite_layer(t, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_TRIANGLE:
        lv_draw_vglite_triangle(t, t->draw_dsc);
        break;
    default:
        break;
}

```

Tutte le operazioni seguenti possono essere eseguite anche con opacità opzionale.

- Riempire l'area con il colore (con raggio o gradiente).
- Eseguire il blit [Block Image Transfer] dell'immagine sorgente (qualsiasi formato da `_vglite_src_cf_supported()`) sulla destinazione (qualsiasi formato da `_vglite_dest_cf_supported()`).
- Ricolorare l'immagine sorgente.
- Ridimensionare e ruotare (90, 180, 270 gradi) l'immagine sorgente RGB565.
- Fusione dei livelli (con gli stessi formati supportati per il blitting).
- Disegnare lettere (lettere bitmap blit / font raster).
- Disegnare bordi completi (LV\_BORDER\_SIDE\_FULL).
- Disegnare archi (con bordi arrotondati).
- Disegnare linee (con trattini o bordi arrotondati).
- Disegnare triangoli colorati (con gradiente).

### Limitazioni note:

- Allineamento dell'immagine sorgente: Il requisito di allineamento dei byte per un pixel dipende dal formato specifico del pixel. Sia l'indirizzo del buffer che il passo del buffer devono essere allineati. Come regola generale, l'allineamento è impostato su 16 pixel. Questo fa sì che l'allineamento dell'indirizzo del buffer sia di 32 byte per RGB565 e di 64 byte per ARGB8888.
- Per la destinazione del pixel engine (PE), l'allineamento dovrebbe essere di 64 byte per tutti i layout di buffer a mosaico (4x4). Il pixel engine non ha requisiti di allineamento aggiuntivi per i layout di buffer lineari (VG\_LITE\_LINEAR).

### Setup del progetto:

- Aggiungere i file sorgente correlati a VGLite (e i gli header corrispondenti, se disponibili) al progetto:
  - "src/draw/nxp/vglite/lv\_draw\_buf\_vglite.c": callback del buffer di disegno
  - "src/draw/nxp/vglite/lv\_draw\_vglite\_arc.c": disegno dell'arco
  - "src/draw/nxp/vglite/lv\_draw\_vglite\_border.c": disegno del bordo
  - "src/draw/nxp/vglite/lv\_draw\_vglite\_fill.c": riempimento dell'area
  - "src/draw/nxp/vglite/lv\_draw\_vglite\_img.c": immagine blit (con ricolorazione o trasformazione facoltativa)
  - "src/draw/nxp/vglite/lv\_draw\_vglite\_label.c": disegno dell'etichetta

- "src/draw/nxp/vglite/lv\_draw\_vglite\_layer.c": fusione livelli
- "src/draw/nxp/vglite/lv\_draw\_vglite\_line.c": disegno linee
- "src/draw/nxp/vglite/lv\_draw\_vglite\_triangle.c": disegna triangoli
- "src/draw/nxp/vglite/lv\_draw\_vglite.c": inizializzazione dell'unità di disegno
- "src/draw/nxp/vglite/lv\_vglite\_buf.c": init/get buffer vglite
- "src/draw/nxp/vglite/lv\_vglite\_matrix.c": imposta la matrice vglite
- "src/draw/nxp/vglite/lv\_vglite\_path.c": crea i dati del percorso vglite
- "src/draw/nxp/vglite/lv\_vglite\_utils.c": helper delle funzioni

## Acceleratore G2D

### Configurazione di base:

- Selezionare il motore NXP G2D in "lv\_conf.h": Impostare **LV\_USE\_G2D** a 1.
- Per utilizzare G2D come unità di disegno, selezionare in "lv\_conf.h": Impostare **LV\_USE\_DRAW\_G2D** a 1.
- Enable G2D asserts in "lv\_conf.h": Set :c:macro: **LV\_USE\_G2D\_ASSERT** to 1. There are few G2D assertions that can stop the program execution in case the :c:macro: **LV\_ASSERT\_HANDLER** is set to **while(1);** (Halt by default). Else, there will be logged just an error message via **LV\_LOG\_ERROR**.

### Inizializzazione di base:

L'inizializzazione del disegno G2D viene eseguita automaticamente in **lv\_init()** una volta che G2D è abilitato come unità di disegno, non è richiesto alcun codice utente:

```
#if LV_USE_DRAW_G2D
    lv_draw_g2d_init();
#endif
```

Durante l'inizializzazione di G2D, verrà creata una nuova unità di disegno **lv\_draw\_g2d\_unit\_t** con le callback aggiuntive, se **LV\_USE\_DRAW\_G2D** è impostato a 1:

```
lv_draw_g2d_unit_t * draw_g2d_unit = lv_draw_create_unit(sizeof(lv_draw_g2d_unit_t));
draw_g2d_unit->base_unit.evaluate_cb = _g2d_evaluate;
draw_g2d_unit->base_unit.dispatch_cb = _g2d_dispatch;
draw_g2d_unit->base_unit.delete_cb = _g2d_delete;
```

e verrà generato un thread aggiuntivo **\_g2d\_render\_thread\_cb()** per gestire le attività di disegno supportate.

```
#if LV_USE_G2D_DRAW_THREAD
    lv_thread_init(&draw_g2d_unit->thread, LV_THREAD_PRIO_HIGH, _g2d_render_thread_cb, 2 * 1024, draw_g2d_unit);
#endif
```

Se **LV\_USE\_G2D\_DRAW\_THREAD** non è definito, non verrà creato alcun thread di disegno aggiuntivo e l'attività di disegno G2D verrà eseguita sullo stesso thread principale LVGL.

**\_g2d\_evaluate()** verrà chiamato dopo la creazione di ogni attività e analizzerà se l'attività è supportata da G2D o meno. Se supportato, all'attività verranno assegnati un punteggio preferito e l'ID dell'unità di disegno. Uno **score** uguale a **100** è lo score di default della CPU. Uno score più piccolo indica che G2D è in grado di disegnarlo più velocemente.

**\_g2d\_dispatch()** è la callback del dispatcher G2D, che prenderà un'attività pronta per essere disegnata (con **DRAW\_UNIT\_ID\_G2D** impostato) e la passerà all'unità di disegno G2D per l'elaborazione.

**\_g2d\_delete()** pulirà l'unità di disegno G2D.

### Funzionalità supportate:

Diverse funzionalità di disegno in LVGL possono essere demandate al motore G2D. La CPU è disponibile per altre operazioni mentre G2D è in esecuzione. È necessario il sistema operativo Linux per bloccare il thread di disegno LVGL e passare a un altro task o sospendere la CPU per risparmiare energia.

Le attività di disegno supportate sono disponibili in "src/draw/nx/g2d/lv\_draw\_g2d.c":

```
switch(t->type) {
    case LV_DRAW_TASK_TYPE_FILL:
        lv_draw_g2d_fill(u, t->draw_dsc, &t->area);
        break;
    case LV_DRAW_TASK_TYPE_IMAGE:
        lv_draw_g2d_img(u, t->draw_dsc, &t->area);
        break;
    default:
        break;
}
```

- Riempire l'area con un colore (senza raggio, senza gradiente) + opacità opzionale.
- Blit [Block Image Transfer] dell'immagine sorgente ARGB8888 sulla destinazione. ARGB8888 + opacità opzionale.
- Ridimensionare l'immagine sorgente ARGB8888.

### Limitazioni note:

- G2D/PXP can only rotate at 90° angles.
- La rotazione non è supportata per le immagini non allineate a blocchi di 16x16 pixel. G2D/PXP è impostato per elaborare blocchi 16x16 per ottimizzare il sistema in termini di larghezza di banda di memoria e tempo di elaborazione delle immagini. Il motore di output essenzialmente tronca tutti i pixel di output dopo che è stato scritto il numero desiderato di pixel. Quando si ruota un'immagine sorgente e l'output non è divisibile per la dimensione del blocco, i pixel errati potrebbero essere troncati e l'immagine di output finale potrebbe apparire spostata.
- La ricolorazione o la trasformazione di immagini con opacità o canale alfa non possono essere ottenute in una singola configurazione della pipeline G2D/PXP. Sarebbero necessari due o più passaggi.
- L'indirizzo del buffer deve essere allineato a 64 byte: impostare `LV_DRAW_BUF_ALIGN` a 64 in "lv\_conf.h". Non è richiesto alcun allineamento di passo: impostare `LV_DRAW_BUF_STRIDE_ALIGN` a 1 in "lv\_conf.h".

### Setup del progetto:

- Aggiungere i file sorgente relativi a G2D (e i relativi header, se disponibili) al progetto:
  - "src/draw/nxp/g2d/lv\_draw\_buf\_g2d.c": callback del buffer di disegno
  - "src/draw/nxp/g2d/lv\_draw\_g2d\_fill.c": riempimento dell'area
  - "src/draw/nxp/g2d/lv\_draw\_g2d\_img.c": immagine blit (con ricolorazione o trasformazione opzionale)
  - "src/draw/nxp/g2d/lv\_draw\_g2d.c": inizializzazione dell'unità di disegno
  - "src/draw/nxp/g2d/lv\_draw\_g2d\_buf\_map.c": mappa hash per i buffer g2d
  - "src/draw/nxp/g2d/lv\_g2d\_utils.c": helper delle funzioni

## 4.4.5 Renesas

### Driver Nativi

Renesas è un partner ufficiale di LVGL. Pertanto, LVGL include il supporto nativo per [Dave2D](#) (la GPU di Renesas) e gestiamo anche progetti Renesas pronti all'uso.

### Dave2D

Dave2D è in grado di accelerare la maggior parte delle operazioni di disegno di LVGL:

- Disegno di rettangoli, anche con gradienti
- Disegno, ridimensionamento e rotazione di immagini
- Disegno di lettere

- Disegno di triangoli
- Disegno di linee

Poiché Dave2D lavora in background, la CPU è libera per altre attività. In pratica, durante il rendering, Dave2D può ridurre l'utilizzo della CPU dalla metà a un terzo, a seconda dell'applicazione.

## GLCDC

GLCDC è una periferica di uscita grafica multistadio disponibile in diversi MCU Renesas. È in grado di pilotare pannelli LCD tramite un'interfaccia RGB altamente configurabile.

Maggiori informazioni sono disponibili nella [pagina dei driver](#).

## Driver MPU

Le MPU Renesas (ovvero la famiglia RZ/G) possono utilizzare questi driver MPU LVGL.

## Wayland

L'utilizzo del driver Wayland è supportato sulle schede madri con un desktop Wayland, ovvero un compositore Wayland.

## fbdev

Il driver video LVGL **fbdev** (frame buffer) può essere utilizzato quasi sempre.

## RA Family

Schede supportate nella RA Family:

- **EK-RA8D1**
- **EK-RA8P1**
- **EK-RA6M3G**

## Esecuzione del Progetto

- L'IDE ufficiale di Renesas si chiama e<sup>2</sup> studio. Essendo basato su Eclipse, funziona anche su Windows, Linux e Mac. I componenti della famiglia RA richiedono determinate versioni minime del Flexible Software Package (FSP). Per informazioni, consultare i singoli file README.md nei repository applicabili. Gli FSP possono essere scaricati [qui](#). Si noti che a partire dalla versione 6.0 di FSP, LVGL fa ora parte della sezione middleware dei componenti. Pertanto, quando si clonano le ultime versioni dei repository delle schede LVGL supportate, non è più necessario l'aggiornamento ricorsivo dei sottomoduli git.
- JLink viene utilizzato per il debug e può essere scaricato [qui](#).
- Clonare il repository pronto all'uso per la scheda selezionata, per versioni FSP precedenti alla 6.0:

```
git clone https://github.com/lvgl/lv_port_renesas_ek-ra8d1.git --recurse-submodules
```

Scaricare il file **.zip** da GitHub non funziona perché non scarica i sottomoduli.

- Clonare il repository pronto all'uso per la scheda selezionata, per versioni FSP dalla 6.0 in poi:

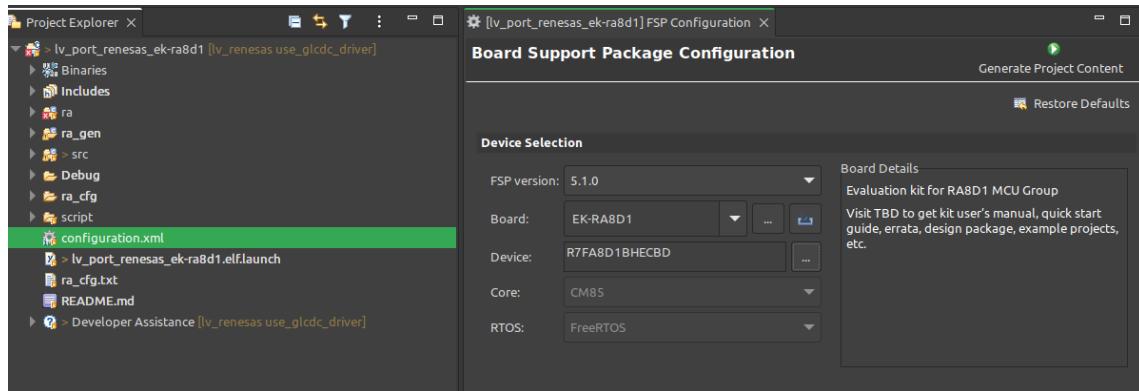
```
git clone https://github.com/lvgl/lv_port_renesas_ek-ra8d1.git
```

- Aprire e<sup>2</sup> studio, andare su **File -> Import project** e selezionare **General/Existing projects into workspace**.
- Selezionare la cartella clonata e premere **Finish**.

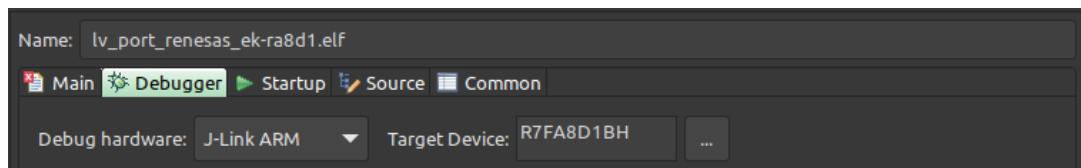
- Doppio-click su `configuration.xml`. Questo attiverà la Configuration Window.

Il Flexible Software Package (FSP) di Renesas include il supporto per i livelli BSP e HAL, esteso a diverse varianti di RTOS e altri stack middleware. I componenti saranno disponibili tramite la generazione di codice, incluso il punto di ingresso in `"main.c"`.

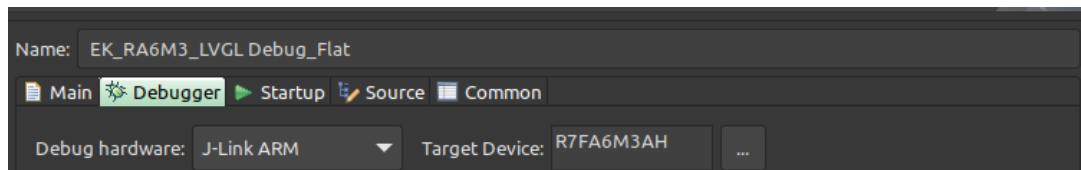
Premere **Generate Project Content** nell'angolo in alto a destra.



- Compilare il progetto premendo **Ctrl + Alt + B**
- Cliccare sul pulsante Debug (  ). Se viene richiesto **Debug Configurations**, sul tab **Debugger** selezionare il **J-Link ARM** come **Debug hardware** e l'**IC** appropriato come **Target Device**:
  - R7FA8D1BH per EK-RA8D1



- R7FA6M3AH per EK-RA6M3G



### Nota

Sulle schede EK-RA8D1, il DIP switch 7 **SW1** (al centro della board) deve essere impostato su ON, tutti gli altri su OFF.

Si noti inoltre che l'apertura di un progetto precedentemente creato su FSP prima della versione 6.0 aprirà una finestra di dialogo che chiederà all'utente se desidera migrare al nuovo FSP v6.0. La migrazione non interromperà il progetto.

## Modificare il progetto

### Aprire una demo

Il punto di ingresso del task principale è contenuto in `src/LVGL_thread_entry.c`.

È possibile disabilitare le demo LVGL (o semplicemente commentarle) e chiamare alcune funzioni `lv_example_...()`, oppure aggiungere codice personalizzato.

## Configurazione

`src/lv_conf.h` (o `src/lv_conf_user.h`) contiene le impostazioni più importanti per LVGL. Ovvero:

- `LV_COLOR_DEPTH` per impostare la profondità di colore predefinita di LVGL
- `LV_MEM_SIZE` per impostare la RAM massima disponibile per LVGL
- `LV_USE_DAVE2D` per abilitare la GPU

I componenti hardware e software possono essere modificati visivamente utilizzando la **Configuration Window**.

As of e<sup>2</sup> Studio 2025-04.1 and FSP 6.0.0, LVGL is a managed component.

## Supporto

In caso di problemi o domande, aprire una "issue" nel repository corrispondente.

## RX Family

Schede supportate nella RX Family:

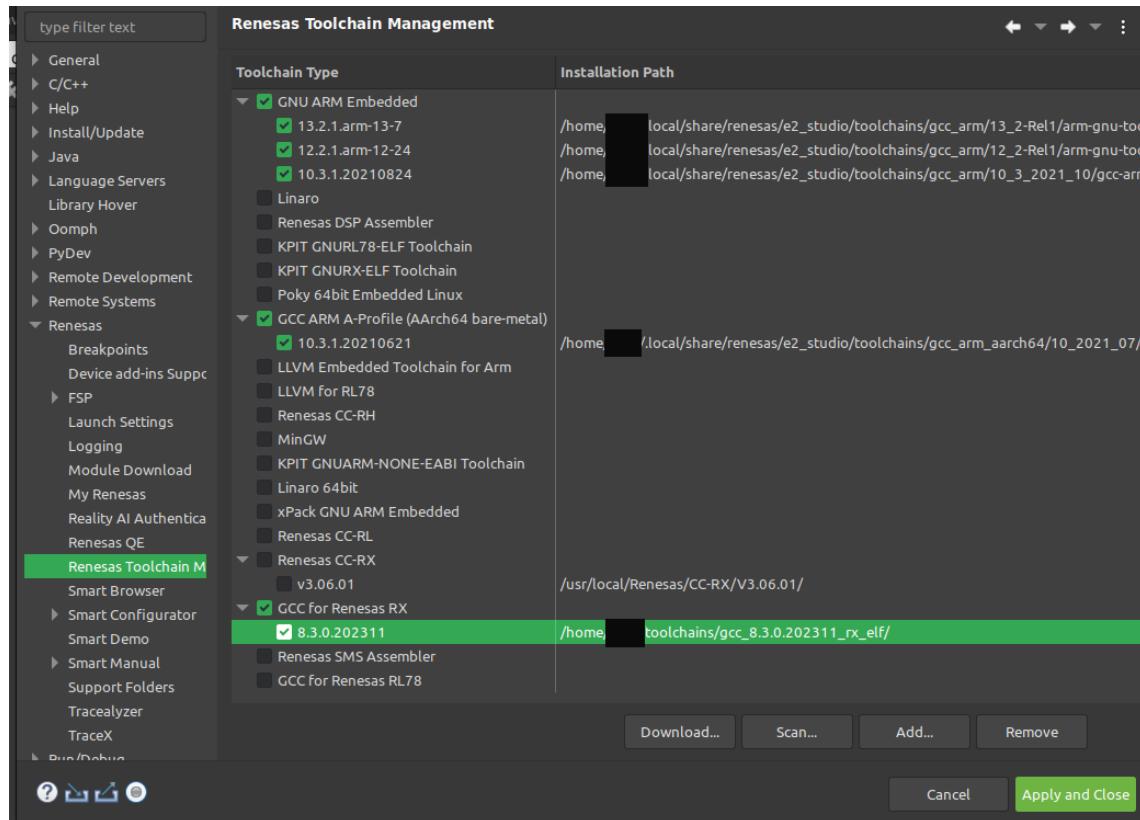
- RX72N Envision Kit

## Esecuzione del Progetto

- L'IDE ufficiale di Renesas si chiama e<sup>2</sup> studio. Essendo basato su Eclipse, funziona anche su Windows, Linux e Mac. Può essere scaricato [qui](#).
- Scaricare e installare il driver necessario per il debugger
  - per Windows: [64 bit](#) [qui](#) e [32 bit](#) [qui](#)
  - per Linux: [qui](#)

Il compilatore deve essere attivato in e<sup>2</sup> studio:

- Si va su **Help -> Add Renesas Toolchains**
- Si preme il pulsante **Add . . .**
- Si seleziona la cartella di installazione della toolchain



- Clonare il repository già pronto `lv_port_renesas_rx72n-envision-kit`:

```
git clone https://github.com/lvgl/lv_port_renesas_rx72n-envision-kit.git --recurse-submodules
```

Scaricare il file .zip da GitHub non funziona perché non scarica i sottomoduli.

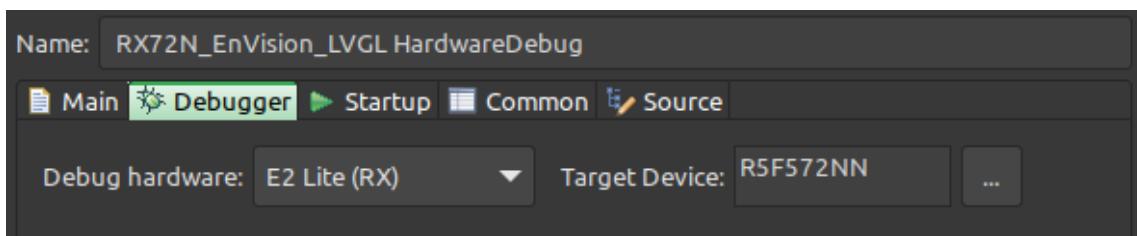
- Aprire e<sup>2</sup> studio, andare su **File -> Import project** e selezionare **General/Existing projects into workspace**
- Selezionare la cartella clonata e premere **Finish**.
- Doppio-click su `RX72N_EnVision_LVGL.scfg` per attivare la **Configuration Window**.

Lo Smart Configurator (SMC) di Renesas include il supporto per i layer BSP e HAL esteso a più varianti RTOS e altri stack middleware. I componenti saranno disponibili tramite la generazione di codice, incluso il punto di ingresso dell'applicazione.

Premere **Generate Code** nell'angolo in alto a destra.

The screenshot shows the Smart Configurator interface. On the left, there's a sidebar with links for Overview, Videos, What's New, Product Documentation, and Current Configuration. The Current Configuration section shows the selected board as R5F572NNhFB and lists components like r\_bsp, r\_cmt\_rx, r\_drw2d\_rx, FreeRTOS\_Kernel, and FreeRTOS\_Object. A central diagram illustrates the software stack: Application Code (RTOS, Middleware & Drivers, Device Drivers) running on MCU Hardware, with Smart Configurator providing configuration data to the Application Code.

- Compilare il progetto premendo **Ctrl + Alt + B**
- Cliccare sul pulsante Debug (  ). Se viene richiesto **Debug Configurations**, sul tab **Debugger** selezionare il **E2 Lite** come **Debug hardware** e **R5F572NN** come **Target Device**:



### Nota

Assicurarsi che entrambi i canali del DIP switch **SW1** (accanto a **ECN1**) siano OFF.

## Modificare il progetto

### Aprire una demo

Il punto di ingresso del task principale è contenuto in `src/LVGL_thread_entry.c`.

È possibile disabilitare le demo LVGL (o semplicemente commentarle) e chiamare alcune funzioni `lv_example_...()`, oppure aggiungere codice personalizzato.

### Configurazione

`src/lv_conf.h` contiene le impostazioni più importanti per LVGL. Ovvero:

- `LV_COLOR_DEPTH` per impostare la profondità di colore predefinita di LVGL
- `LV_MEM_SIZE` per impostare la RAM massima disponibile per LVGL
- `LV_USE_DAVE2D` per abilitare la GPU

I componenti hardware e software possono essere modificati visivamente utilizzando la Configuration Window.

## Supporto

In caso di problemi o domande, aprire una "issue" nel repository corrispondente.

## RZ/G Family

Schede supportate nella RZ/G Family:

- **RZ/G2L-EVKIT**
- **RZ/G2UL-EVKIT**

## Esecuzione del Progetto

- Le schede RZ/G sono MPU con supporto per Linux. I progetti vengono sviluppati utilizzando un SDK con toolchain CLI cross-compilatore in un ambiente PC Linux (WSL, Docker, ecc. possono essere utilizzati su Windows).
- La G2L ha un desktop Wayland e il progetto appare come una finestra Wayland. La G2UL non ha un desktop, quindi il progetto è a schermo intero e utilizza il driver `fbdev`.
- L'SDK attualmente utilizza LVGL v8.3, quindi questo progetto utilizza questa versione per rispecchiare la versione dell'SDK, anche se LVGL è linkato staticamente. È possibile provare a utilizzare versioni più recenti di LVGL. Consultare la [v8-to-v9 Migration Guide](#) per gli aspetti da considerare.
- Clonare il repository pronto all'uso per la board selezionata:

```
git clone https://github.com/lvgl/lv_port_renesas_rz-g2l-evkit --recurse-submodules
```

Scaricare il file `.zip` da GitHub non funziona perché non scarica i sottomoduli.

- Seguire le istruzioni nel file README.md del progetto per ottenere l'immagine della scheda SD e il programma di installazione della toolchain, compila e carica il progetto sulla scheda.
- Interrompere eventuali demo avviate automaticamente (su G2UL eseguire `systemctl stop demo-launcher` nel terminale).
- Esecuzione del progetto

```
./lvgl_demo_benchmark
```

## Modificare il progetto

### Aprire una demo

Il punto di ingresso è contenuto in `src/main.c`.

È possibile disabilitare le demo LVGL (`lv_demo_benchmark()`) (o semplicemente commentarle) e chiamare alcune funzioni `lv_example_...()`, oppure aggiungere codice personalizzato.

### Configurazione

Modificare `lv_conf.h` e `lv_drv_conf.h` per configurare LVGL. L'immagine della board contiene LVGL e `lv_drivers` come librerie linkabili dinamicamente. Questo progetto compila LVGL staticamente per la personalizzazione e per il porting del benchmark LVGL v9 da LVGL v8.3.

## Supporto

In caso di problemi o domande, aprire una "issue" nel repository corrispondente.

## RZ/A Family

Schede supportate nella RZ/A Family:

- **RZ/A3M**

## Esecuzione del Progetto

- The RZ/A boards are MPUs with a focus on bare metal and RTOS applications. Projects are built for them using e<sup>2</sup> Studio IDE, available for Windows, Mac, and Linux.
- Clonare il repository pronto all'uso per la propria board:

```
git clone https://github.com/lvgl/lv_port_renesas-ek-rz_a3m --recurse-submodules
```

Scaricare il file `.zip` da GitHub non funziona perché non scarica i sottomoduli.

- Seguire le istruzioni nel file README.md del progetto per compilare e flashare il progetto sulla board.

## Modificare il progetto

### Aprire una demo

Il punto di ingresso è contenuto in `src/LVGL_thread_entry.c`.

È possibile disabilitare le demo LVGL (`lv_demo_benchmark()`) (o semplicemente commentarle) e chiamare alcune funzioni `lv_example_...()`, oppure aggiungere codice personalizzato.

### Configurazione

Modificare `lv_conf.h` per configurare LVGL.

Eseguirà automaticamente qualsiasi demo abilitata in `lv_conf.h`. È possibile consultare `lv_conf.defaults` per un riepilogo delle configurazioni modificate rispetto a quelle di default.

## Supporto

In caso di problemi o domande, aprire una "issue" nel repository corrispondente.



## Board Supportate

Nome della Board	CPU	Memoria	Display	Board video	Link
<b>EK-RA8D1</b>					Repository delle demo
	480 MHz Arm Cortex-M85	1MB SDRAM interna 64MB SDRAM esterna  2MB flash interna  64MB octo-SPI flash esterna	4.5" 480x854 2-lane MIPI		
<b>EK-RA8P1</b>					Repository demo
	1 GHz Arm Cortex-M85	1872KB interna 128MB esterna 1MB flash interna	7" 1024x600 Parallel RGB888		
<b>EK-RA6M3G</b>					Repository demo
	120 MHz Arm Cortex-M4	640kB SRAM interna 2MB SRAM interna  32MB QSPI flash esterna	4.3" 480x272 Parallel RGB565		
<b>RX72N Envision Kit</b>					Repository demo
	240 MHz Renesas RXv3	1MB SRAM interna 4MB SRAM interna  32MB QSPI flash esterna	4.3" 480x272 Parallel RGB565		
<b>RZ/G2L-EVKIT</b>					Repository demo
	1.2 GHz Arm Cortex-A55	2GB DDR4 SDRAM	1920x1080 Micro-HDMI		
<b>RZ/G2UL-EVK</b>					Repository demo
	1.0 GHz Arm Cortex-A55	1GB DDR4 SDRAM	1280x800 Parallel RGB888 con Micro-HDMI		
<b>RZ/A3M</b>					Repository demo
	1.0 GHz Arm Cortex-A55	128MB DDR3L-SDRAM interna 128KB SRAM	4.3" 1280x720 MIPI		
<b>4.4. Fornitori di chip</b>		interna 128MB QSPI NAND flash esterna 32MB QSPI			158

## 4.4.6 STM32

LVGL può essere aggiunto a STM32CubeIDE in modo simile a qualsiasi altro IDE basato su Eclipse.

### Inclusione di LVGL in un Progetto

- Creare o aprire un progetto in STM32CubeIDE.
- Copiare l'intera cartella LVGL in *[project\_folder]/Drivers/lvgl*.
- Nel riquadro **Project Explorer** di STM32CubeIDE: cliccare col pulsante destro del mouse sulla cartella LVGL copiata (potrebbe essere necessario aggiornare la vista prima che venga visualizzata) e selezionare **Add/remove include path....** Se questa opzione non viene visualizzata o non funziona, si può controllare i path di inclusione del progetto nel menu **Project -> Properties**, quindi andare a **C/C++ Build -> Settings -> Include paths** e assicurarsi che la directory LVGL sia elencata.

Ora che i file sorgente sono inclusi nel progetto, seguire le istruzioni per [aggiungere LVGL al progetto](#) e creare il file `lv_conf.h`, quindi inizializzare il display.

### Esempio Bare Metal

Un esempio minimale che utilizza STM32CubeIDE e HAL. \* Quando si imposta **Pinout and Configuration** utilizzando il **Device Configuration Tool**, selezionare **System Core -> SYS** e assicurarsi che **Timebase Source** sia impostato su **SysTick**. \* Configurare eventuali altre periferiche (incluso il pannello LCD) e inizializzarle in *main.c*. \* `#include "lvgl.h"` nel file *main.c*. \* Creare alcuni frame buffer come variabili globali:

```
/* Frame buffers
 * Static or global buffer(s). The second buffer is optional
 * TODO: Adjust color format and choose buffer size. DISPLAY_WIDTH * 10 is one suggestion. */
#define BYTES_PER_PIXEL (LV_COLOR_FORMAT_GET_SIZE(LV_COLOR_FORMAT_RGB565)) /* will be 2 for RGB565 */
#define BUFF_SIZE (DISPLAY_WIDTH * 10 * BYTES_PER_PIXEL)
static uint8_t buf_1[BUFF_SIZE];
static uint8_t buf_2[BUFF_SIZE];
```

- Nella funzione `main()`, dopo aver inizializzato la CPU, le periferiche e il pannello LCD, chiamare `lv_init()` per inizializzare LVGL. Si può poi creare il driver video usando `lv_display_create()` e registrare i frame buffer usando `lv_display_set_buffers()`.

```
//Initialise LVGL UI library
lv_init();

lv_display_t * disp = lv_display_create(WIDTH, HEIGHT); /* Basic initialization with horizontal and vertical resolution in pixels */
lv_display_set_flush_cb(disp, my_flush_cb); /* Set a flush callback to draw to the display */
lv_display_set_buffers(disp, buf_1, buf_2, sizeof(buf_1), LV_DISPLAY_RENDER_MODE_PARTIAL); /* Set an initialized buffer */
```

- Creare alcuni widget fittizi per testare l'output:

```
/* Change Active Screen's background color */
lv_obj_set_style_bg_color(lv_screen_active(), lv_color_hex(0x003a57), LV_PART_MAIN);
lv_obj_set_style_text_color(lv_screen_active(), lv_color_hex(0xffffffff), LV_PART_MAIN);

/* Create a spinner */
lv_obj_t * spinner = lv_spinner_create(lv_screen_active());
lv_obj_set_size(spinner, 64, 64);
lv_obj_align(spinner, LV_ALIGN_BOTTOM_MID, 0, 0);
```

- Aggiungere una chiamata a `lv_timer_handler()` all'interno del ciclo `while(1)`:

```
/* Infinite loop */
while (1)
{
    lv_timer_handler();
    HAL_Delay(5);
}
```

- Aggiungere una chiamata a `lv_tick_inc()` all'interno della funzione `SysTick_Handler()`. Aprire il file *stm32xxxx\_it.c* (il nome dipenderà dalla propria MCU specifica) e aggiornare la funzione `SysTick_Handler()`:

```
void SysTick_Handler(void)
{
    /* USER CODE BEGIN SysTick_IRQn 0 */

    HAL_SYSTICK_IRQHandler();
    lv_tick_inc(1);
    #ifdef USE_RTTOS_SYSTICK
        osSystickHandler();
    #endif
}
```

(continues on next page)

(continua dalla pagina precedente)

```

#endif

/* USER CODE END SysTick_IRQn 0 */
HAL_IncTick();
/* USER CODE BEGIN SysTick_IRQn 1 */

/* USER CODE END SysTick_IRQn 1 */
}

```

- Infine, scrivere la funzione di callback, `my_flush_cb`, che invierà il buffer del display al pannello LCD. Di seguito è riportato un esempio, ma varierà a seconda della configurazione.

```

void my_flush_cb(lv_display_t * disp, const lv_area_t * area, lv_color_t * color_p)
{
    //Set the drawing region
    set_draw_window(area->x1, area->y1, area->x2, area->y2);

    int height = area->y2 - area->y1 + 1;
    int width = area->x2 - area->x1 + 1;

    //We will do the SPI write manually here for speed
    HAL_GPIO_WritePin(DC_PORT, DC_PIN, GPIO_PIN_SET);
    //CS low to begin data
    HAL_GPIO_WritePin(CS_PORT, CS_PIN, GPIO_PIN_RESET);

    //Write colour to each pixel
    for (int i = 0; i < width * height; i++) {
        uint16_t color_full = (color_p->red << 11) | (color_p->green << 5) | (color_p->blue);
        parallel_write(color_full);

        color_p++;
    }

    //Return CS to high
    HAL_GPIO_WritePin(CS_PORT, CS_PIN, GPIO_PIN_SET);

    /* IMPORTANT!!!
     * Inform the graphics library that you are ready with the flushing */
    lv_display_flush_ready(disp);
}

```

## Esempio FreeRTOS

Un esempio minimo che utilizza STM32CubeIDE, HAL e CMSISv1 (FreeRTOS). Notare che non abbiamo utilizzato i mutex in questo esempio, tuttavia LVGL \*\*\*NON\*\*\* è "thread safe" e quindi è consigliabile utilizzare i mutex. Vedere: *Considerazioni sui Thread* \* #include "lvgl.h" \* Create your frame buffer(s) as global variables:

```

/* Frame buffers
 * Static or global buffer(s). The second buffer is optional */
#define BYTES_PER_PIXEL (LV_COLOR_FORMAT_GET_SIZE(LV_COLOR_FORMAT_RGB565)) /* will be 2 for RGB565 */
/* TODO: Declare your own BUFF_SIZE appropriate to your system. */
#define BUFF_SIZE (DISPLAY_WIDTH * 10 * BYTES_PER_PIXEL)
static uint8_t buf_1[BUFF_SIZE];
static uint8_t buf_2[BUFF_SIZE];

```

- Nella funzione `main`, dopo aver inizializzato le periferiche (SPI, GPIO, LCD ecc.), inizializzare LVGL usando `lv_init()`, creare un nuovo driver video usando `lv_display_create()` e registrare i frame buffer usando `lv_display_set_buffers()`.

```

/* Initialise LVGL UI library */
lv_init();
lv_display_t *display = lv_display_create(WIDTH, HEIGHT); /* Create the display */
lv_display_set_flush_cb(display, my_flush_cb); /* Set a flush callback to draw to the display */
lv_display_set_buffers(display, buf_1, buf_2, sizeof(buf_1), LV_DISPLAY_RENDER_MODE_PARTIAL); /* Set an initialized buffer */

/* Register the touch controller with LVGL - Not included here for brevity. */

```

- Creare alcuni widget fitizi per testare l'output:

```

/* Change Active Screen's background color */
lv_obj_set_style_bg_color(lv_screen_active(), lv_color_hex(0x003a57), LV_PART_MAIN);
lv_obj_set_style_text_color(lv_screen_active(), lv_color_hex(0xffffffff), LV_PART_MAIN);

/* Create a spinner */
lv_obj_t * spinner = lv_spinner_create(lv_screen_active());
lv_obj_set_size(spinner, 64, 64);
lv_obj_align(spinner, LV_ALIGN_BOTTOM_MID, 0, 0);

```

- Creare due thread per chiamare `lv_timer_handler()` e `lv_tick_inc()`. Ci sarà bisogno di due handle `osThreadId` per CMSISv1. In questo caso, non è necessario che siano accessibili globalmente, tuttavia la generazione del codice STM32Cube lo fa di default. Se si usa CMSIS e la generazione del codice STM32Cube, il risultato dovrebbe essere simile a questo:

```

//Thread Handles
osThreadId lvgl_tickHandle;
osThreadId lvgl_timerHandle;

/* definition and creation of lvgl tick */
osThreadDef(lvgl_tick, LVGLTick, osPriorityNormal, 0, 1024);

```

(continues on next page)

(continua dalla pagina precedente)

```
lvgl_tickHandle = osThreadCreate(osThread(lvgl_tick), NULL);
//LVGL update timer
osThreadDef(lvgl_timer, LVGLTimer, osPriorityNormal, 0, 1024);
lvgl_timerHandle = osThreadCreate(osThread(lvgl_timer), NULL);
```

- E creare le funzioni del thread:

```
/* LVGL timer for tasks. */
void LVGLTimer(void const * argument)
{
    for(;;)
    {
        lv_timer_handler();
        osDelay(20);
    }
}
/* LVGL tick source */
void LVGLTick(void const * argument)
{
    for(;;)
    {
        lv_tick_inc(10);
        osDelay(10);
    }
}
```

- Infine, creare la funzione `my_flush_cb` per inviare il frame buffer all'LCD. Le specifiche di questa funzione variano a seconda delle funzionalità della MCU in uso. Di seguito è riportato un semplice esempio di interfaccia LCD parallela; adattarla in base alle specifiche capacità del proprio display e della MCU.

```
void my_flush_cb(lv_display_t * display, const lv_area_t * area, uint8_t * px_map)
{
    uint16_t * color_p = (uint16_t *)px_map;

    //Set the drawing region
    set_draw_window(area->x1, area->y1, area->x2, area->y2);

    int height = area->y2 - area->y1 + 1;
    int width = area->x2 - area->x1 + 1;

    //Begin SPI Write for DATA
    HAL_GPIO_WritePin(DC_PORT, DC_PIN, GPIO_PIN_SET);
    HAL_GPIO_WritePin(CS_PORT, CS_PIN, GPIO_PIN_RESET);

    //Write colour to each pixel
    for (int i = 0; i < width * height; i++) {
        parallel_write(color_p);
        color_p++;
    }

    //Return CS to high
    HAL_GPIO_WritePin(CS_PORT, CS_PIN, GPIO_PIN_SET);

    /* IMPORTANT!!!
     * Inform the graphics library that you are ready with the flushing */
    lv_display_flush_ready(display);
}
```

## Supporto DMA2D

LVGL supporta DMA2D, una funzionalità di alcuni MCU STM32 che può migliorare le prestazioni durante i riempimenti "blending" e immagini. Alcune linee di prodotti STM32, come STM32F4, STM32F7, STM32L4, STM32U5 e STM32H7, includono modelli con supporto DMA2D.

L'integrazione di LVGL con DMA2D può essere abilitata impostando `LV_USE_DRAW_DMA2D` a 1 in `lv_conf.h`

Con `LV_USE_DRAW_DMA2D_INTERRUPT` impostato a 0 e `LV_USE_OS` impostato a `LV_OS_NONE`, DMA2D disegnerà alcuni riempimenti e immagini contemporaneamente al rendering software, ove possibile. Se `LV_USE_DRAW_DMA2D_INTERRUPT` è impostato a 1 e `LV_USE_OS` impostato a `LV_OS_FREERTOS` (o un altro sistema operativo), la differenza principale sarà che il core rimarrà inattivo invece di "busywait" in attesa del completamento di un trasferimento DMA2D.

Se `LV_USE_DRAW_DMA2D_INTERRUPT` è abilitato, è necessario chiamare `lv_draw_dma2d_transfer_complete_interrupt_handler` ogni volta che viene ricevuto l'interrupt globale DMA2D "transfer complete".

DMA2D consente inoltre di combinare layer con formato colore su `LV_COLOR_FORMAT_ARGB1555` sopra i layer `LV_COLOR_FORMAT_RGB565`.

Se il dispositivo STM ha una GPU NeoChrom, si può usare invece il *Nema GFX renderer*.

## 4.5 Driver

### 4.5.1 Display

#### Driver Framebuffer Linux

##### Panoramica

Il framebuffer Linux (fbdev) è un sottosistema Linux utilizzato per visualizzare la grafica. Si tratta di un'API indipendente dall'hardware che consente al software nello spazio utente di accedere al framebuffer (la parte della memoria video di un computer contenente un frame video corrente) utilizzando solo le funzionalità di base del kernel Linux e la sua interfaccia del file system del dispositivo, evitando la necessità di librerie che implementino i driver video nello spazio utente.

##### Prerequisiti

Il sistema dispone di un dispositivo framebuffer configurato (solitamente in `/dev/fb0`).

##### Configurazione del driver

Abilitare il supporto del driver framebuffer in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig. È inoltre possibile configurare la modalità di rendering.

```
#define LV_USE_LINUX_FBDEV 1
#define LV_LINUX_FBDEV_RENDER_MODE LV_DISPLAY_RENDER_MODE_PARTIAL
```

##### Utilizzo

Per impostare un display basato su framebuffer, creare prima un display con `lv_linux_fbdev_create`. Successivamente, si imposta il nodo del dispositivo framebuffer sul display (di solito è `/dev/fb0`).

```
lv_display_t *disp = lv_linux_fbdev_create();
lv_linux_fbdev_set_file(disp, "/dev/fb0");
```

Se lo schermo rimane nero o disegna solo parzialmente, si può provare ad abilitare il rendering diretto tramite `LV_DISPLAY_RENDER_MODE_DIRECT`. Inoltre, si può attivare una modalità di aggiornamento forzato con `lv_linux_fbdev_set_force_refresh(true)`. Questo di solito ha un impatto sulle prestazioni e non dovrebbe essere abilitato a meno che non sia realmente necessario.

##### Nascondere il cursore

Si potrebbe vedere un cursore lampeggiante sullo schermo. Il metodo per nasconderlo varia a seconda della piattaforma. Ad esempio, ecco come si può fare su un Raspberry Pi:

1. Modificare il file `/boot/cmdline.txt`.
2. Aggiungere `vt.global_cursor_default=0`.

##### Errori comuni

###### Problema della risoluzione di default

Quando il kernel Linux si inizializza, imposta sottosistemi come il framebuffer (fbdev) per gestire l'output del display. Se è collegato un display HDMI, il kernel lo rileva e alloca una porzione di RAM come framebuffer, che contiene i dati pixel per il rendering delle immagini.

Tuttavia, un problema comune si verifica quando il display non viene acceso durante il processo di avvio. Se la scheda madre viene accesa prima dello schermo, il kernel potrebbe configurare erroneamente la risoluzione del framebuffer. Di conseguenza, sia la risoluzione visibile che quella virtuale possono essere errate, causando problemi di visualizzazione.

Questo problema si verifica spesso con le connessioni HDMI in cui il display viene acceso dopo che il sistema si è già avviato.

Il seguente comando restituisce le impostazioni correnti del dispositivo framebuffer specificato, come risoluzione, profondità di pixel e timing.

```
fbset -fb /dev/fb0
```

Per evitare problemi relativi al display, si consiglia di assicurarsi che tutti i dispositivi, incluso il display HDMI, siano collegati e accesi prima di accendere la scheda madre.

## FT81x

Un driver framebuffer minimale per i controller smart display EVE FT81x. Funziona anche per BT81x.

Al momento è supportata la modalità di rendering parziale a buffer singolo Al momento potrebbe non supportare DSPI o QSPI.

Testato a 32 MHz su FT812 e a 23 MHz su BT817.

## Utilizzo

```
#define FB_SIZE          800 * 2 * 50
#define MAX_TRANSFER_SIZE  FB_SIZE

static void spi_cb(lv_display_t * disp, lv_ft81x_spi_operation operation, void * data, uint32_t length)
{
    spi_device_handle_t spi = lv_ft81x_get_user_data(disp);
    switch(operation) {
        case LV_FT81X_SPI_OPERATION_CS_ASSERT:
            gpio_set_level(CS_PIN, 0);
            break;
        case LV_FT81X_SPI_OPERATION_CS_DEASSERT:
            gpio_set_level(CS_PIN, 1);
            esp_rom_delay_us(10); /* tiny delay in case a CS_ASSERT immediately follows */
            break;
        case LV_FT81X_SPI_OPERATION_SEND:
            spi_transaction_t trans = {0};
            while(length) {
                uint32_t sz = length < MAX_TRANSFER_SIZE ? length : MAX_TRANSFER_SIZE;
                trans.length = sz * 8;
                trans.rxlengt

```

(continues on next page)

(continua dalla pagina precedente)

```

.pclk = EVE_PCLK,
.has_crystal = true,
.is_bt81x = false
};

static uint8_t fb[FB_SIZE] __attribute__((aligned(4)));
lv_display_t * disp = lv_ft81x_create(&params, fb, FB_SIZE, spi_cb, spi);
/* ... */
}

```

## Risoluzione dei problemi

Se il "backlight" [retroilluminazione] non si accende (o è troppo luminoso o troppo fioco), provare a modificare il valore di `PWM_DUTY_BACKLIGHT_ON` in `lv_ft81x.c`, che può variare a seconda della board.

## Driver generico per controller LCD compatibile con MIPI DCS

### Panoramica

Da Wikipedia:

**MIPI Alliance** è un'alleanza commerciale globale che sviluppa specifiche tecniche per l'ecosistema mobile, in particolare per gli smartphone, ma include anche i settori influenzati dalla tecnologia mobile.

MIPI è stata fondata nel 2003 da Arm, Intel, Nokia, Samsung, STMicroelectronics e Texas Instruments.

MIPI Alliance ha pubblicato una serie di specifiche relative ai dispositivi di visualizzazione, tra cui DBI (Display Bus Interface), DSI (Display Serial Interface) e DCS (Display Command Set). Di solito, quando si parla di un display compatibile con MIPI, si pensa a un dispositivo con interfaccia seriale DSI. Tuttavia, la specifica Display Bus Interface include una serie di altre interfacce legacy, come l'interfaccia seriale SPI o l'interfaccia parallela compatibile con i8080, che vengono spesso utilizzate per interfacciare i display LCD con microcontrollori di fascia bassa. Inoltre, la specifica DCS contiene un set di comandi standard, supportato da un gran numero di controller LCD TFT legacy, inclusi i popolari SOC Sitronix (ST7735, ST7789, ST7796) e Ilitek (ILI9341). Questi comandi forniscono un'interfaccia comune per configurare l'orientamento del display, la risoluzione del colore, varie modalità di alimentazione e forniscono l'accesso generico alla memoria video. Oltre a questo set di comandi standard, ogni chip controller LCD dispone di una serie di comandi specifici del fornitore per configurare i livelli del generatore di tensione, le temporizzazioni o le curve gamma.

### **i** Nota

È importante comprendere che questo driver LCD MIPI generico non è un driver hardware per display con interfaccia seriale DSI ("MIPI"). Invece, implementa il set di comandi MIPI DCS utilizzato in molti controller LCD con bus SPI o i8080 e fornisce un framework comune per i controller display specifici per chip.

### **💡 Suggerimento**

Sebbene si tratti di un driver generico, può essere utilizzato per supportare chip compatibili che non dispongono di un driver specifico.

## Prerequisiti

Non ci sono prerequisiti.

## Configurazione del driver

Abilitare il supporto del driver LCD MIPI generico in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig.

```
#define LV_USE_GENERIC_MIPI 1
```

**i Nota**

`LV_USE_GENERIC_MIPI` viene abilitato automaticamente quando viene abilitato un driver compatibile.

**Utilizzo**

È necessario implementare due funzioni dipendenti dalla piattaforma:

```
/* Send short command to the LCD. This function shall wait until the transaction finishes. */
int32_t my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    ...
}

/* Send large array of pixel data to the LCD. If necessary, this function has to do the byte-swapping. This function can do the transfer in the background. */
int32_t my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    ...
}
```

L'unica differenza tra le funzioni `my_lcd_send_cmd()` e `my_lcd_send_color()` è che `my_lcd_send_cmd()` viene utilizzata per inviare comandi brevi e ci si aspetta che completi la transazione al suo ritorno (in altre parole, dovrebbe essere bloccante), mentre `my_lcd_send_color()` viene utilizzata solo per inviare dati pixel ed è consigliabile utilizzare DMA per trasmettere i dati in background. È possibile implementare anche metodi più sofisticati, come l'accodamento dei trasferimenti e la loro pianificazione in background.

Si noti che, sebbene il flushing del display sia gestito dal driver, è responsabilità dell'utente chiamare `lv_display_flush_ready()` al termine del trasferimento del colore. In caso di trasferimento DMA, questa operazione viene solitamente eseguita in una callback di "transfer ready".

**i Nota**

Sebbene sia accettabile utilizzare un'implementazione bloccante anche per il trasferimento dei pixel, le prestazioni ne risentiranno.

**💡 Suggerimento**

È necessario prestare attenzione a non inviare un comando mentre è in corso un trasferimento attivo in background. È responsabilità dell'utente implementare questa funzionalità interrogando l'hardware, interrogando una variabile globale (che viene reimpostata al termine del trasferimento) o utilizzando un semaforo o un altro meccanismo di blocco.

Si noti inoltre che il driver non gestisce l'allocazione del "draw buffer" [buffer di disegno], poiché anche questa potrebbe dipendere dalla piattaforma. Pertanto, è necessario allocare i buffer e assegnarli all'oggetto display come di consueto chiamando `lv_display_set_buffers()`.

Il driver può essere utilizzato per creare più display. In una configurazione di questo tipo, le callback devono essere in grado di distinguere tra i display. In genere si implementa un set separato di callback per ciascun display. Si noti inoltre che l'utente deve occuparsi dell'arbitraggio del bus quando più dispositivi sono collegati ad esso.

**Esempio****i Nota**

È possibile trovare una guida passo passo e l'implementazione effettiva dei callback su un STM32F746 utilizzando STM32CubeIDE e le librerie ST HAL qui: [Guida passo passo: come utilizzare i driver LCD LVGL v9 con i dispositivi STM32](#)

```
#include "src/drivers/display/st7789/lv_st7789.h"

#define LCD_H_RES          240
#define LCD_V_RES          320
#define LCD_BUF_LINES      60

lv_display_t *my_disp;
...

/* Initialize LCD I/O bus, reset LCD */
static int32_t my_lcd_io_init(void)
{
    ...
    return HAL_OK;
}

/* Send command to the LCD controller */
static void my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    ...
}

/* Send pixel data to the LCD controller */
static void my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    ...

    /* This must be called to signal that the transfer has finished.
     * It is typically called in a "DMA transfer complete" callback
     * long after `my_lcd_send_color` has returned.
     */
    lv_display_flush_ready(disp);
}

int main(int argc, char ** argv)
{
    ...

    /* Initialize LVGL */
    lv_init();

    /* Initialize LCD bus I/O */
    if (my_lcd_io_init() != 0)
        return;

    /* Create the LVGL display object and the LCD display driver */
    my_disp = lv_lcd_generic_mipi_create(LCD_H_RES, LCD_V_RES, LV_LCD_FLAG_NONE, my_lcd_send_cmd, my_lcd_send_color);

    /* Set display orientation to landscape */
    lv_display_set_rotation(my_disp, LV_DISPLAY_ROTATION_90);

    /* Configure draw buffers, etc. */
    uint8_t * buf1 = NULL;
    uint8_t * buf2 = NULL;

    uint32_t buf_size = LCD_H_RES * LCD_BUF_LINES * lv_color_format_get_size(lv_display_get_color_format(my_disp));

    buf1 = lv_malloc(buf_size);
    if(buf1 == NULL) {
        LV_LOG_ERROR("display draw buffer malloc failed");
        return;
    }
    /* Allocate secondary buffer if needed */
    ...

    lv_display_set_buffers(my_disp, buf1, buf2, buf_size, LV_DISPLAY_RENDER_MODE_PARTIAL);

    ui_init(my_disp);

    while(true) {
        ...

        /* Periodically call the lv_timer handler */
        lv_timer_handler();
    }
}
```

## Argomenti avanzati

### Creazione di flag

Il terzo argomento della funzione `lv_lcd_generic_mipi_create()` è un array di flag. Questo può essere utilizzato per configurare l'orientamento e l'ordinamento RGB del pannello se le impostazioni di default non funzionano. In particolare, il driver MIPI generico accetta i seguenti flag:

LV_LCD_FLAG_NONE
LV_LCD_FLAG_MIRROR_X
LV_LCD_FLAG_MIRROR_Y
LV_LCD_FLAG_BGR

È possibile passare più flag combinandoli tramite OR, ad esempio, `LV_LCD_FLAG_MIRROR_X | LV_LCD_FLAG_BGR`.

## Elenchi di comandi personalizzati

Sebbene i driver specifici per chip facciano del loro meglio per inizializzare correttamente il controller LCD, è possibile che diversi pannelli TFT richiedano configurazioni diverse. In particolare, una corretta impostazione della gamma è fondamentale per una buona riproduzione dei colori. Purtroppo, trovare un buon set di parametri non è facile. Di solito il produttore del pannello fornisce un codice di esempio con le impostazioni di registro consigliate.

È possibile utilizzare la funzione `my_lcd_send_cmd()` per inviare un comando arbitrario al controller LCD. Tuttavia, per semplificare l'invio di un numero elevato di parametri, il driver MIPI generico supporta l'invio di un elenco di comandi personalizzati al controller. I comandi devono essere inseriti in un array di `'uint8_t'`:

```
static const uint8_t init_cmd_list[] = {
    <command 1>, <number of parameters>, <parameter 1>, ... <parameter N>,
    <command 2>, <number of parameters>, <parameter 1>, ... <parameter N>,
    ...
    LV_LCD_CMD_DELAY_MS, LV_LCD_CMD_EOF           /* terminate list: this is required! */
};

...
lv_lcd_generic_mipi_send_cmd_list(my_disp, init_cmd_list);
```

È possibile aggiungere un ritardo tra i comandi utilizzando lo pseudo-comando `LV_LCD_CMD_DELAY_MS`, seguito dal ritardo specificato in unità di 10 ms. Per terminare l'elenco dei comandi, è necessario utilizzare un ritardo con valore `LV_LCD_CMD_EOF`, come mostrato sopra.

Vedere un esempio concreto di invio di un elenco di comandi [qui](#).

## Driver per controller LCD ILI9341

### Panoramica

[ILI9341](#) è un driver SOC a chip singolo da 262.144 colori per display a cristalli liquidi a-TFT con risoluzione di 240 RGB x 320 punti, composto da un driver sorgente a 720 canali, un driver di gate a 320 canali, 172.800 byte di GRAM per dati di visualizzazione grafica di 240 RGB x 320 punti e un circuito di alimentazione. ILI9341 supporta l'interfaccia MCU parallela con bus dati a 8/9/16/18 bit, l'interfaccia RGB con bus dati a 6/16/18 bit e l'interfaccia periferica seriale (SPI) a 3/4 linee.

Il `driver` del controller LCD ILI9341 è un driver indipendente dalla piattaforma, basato sul `drive` generico [MIPI](#). Implementa l'inizializzazione del display, supporta la rotazione del display e implementa il callback di flush del display. L'utente deve implementare solo due funzioni specifiche della piattaforma per inviare un comando o dati pixel al controller tramite SPI o bus parallelo. In genere, queste vengono implementate richiamando le funzioni della libreria SDK appropriate sulla piattaforma specificata.

### Prerequisiti

Non ci sono prerequisiti.

### Configurazione del driver

Abilitare il supporto del driver ILI9341 in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite `KConfig`.

```
#define LV_USE_ILI9341 1
```

### Utilizzo

È necessario implementare due funzioni dipendenti dalla piattaforma:

```
/* Send short command to the LCD. This function shall wait until the transaction finishes. */
int32_t my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    ...
}

/* Send large array of pixel data to the LCD. If necessary, this function has to do the byte-swapping. This function can do the transfer in the background. */
int32_t my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    ...
}
```

Per creare un display basato su ILI9341, utilizzare la funzione

```
/*
 * Create an LCD display with ILI9341 driver
 * @param hor_res      horizontal resolution
 * @param ver_res      vertical resolution
 * @param flags        default configuration settings (mirror, RGB ordering, etc.)
 * @param send_cmd     platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
 * @param send_color   platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' hook)
 * @param callback     pointer to the created display
 */
lv_display_t * lv_ilis9341_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                  lv_ilis9341_send_cmd_cb_t send_cmd_cb, lv_ilis9341_send_color_cb_t send_color_cb);
```

Per ulteriori dettagli e un esempio pratico, consultare la documentazione generica del driver MIPI.

### i Nota

È possibile trovare una guida passo passo e l'implementazione effettiva dei callback su un STM32F746 utilizzando STM32CubeIDE e le librerie ST HAL qui: [Guida passo passo: come utilizzare i driver LCD LVGL v9 con i dispositivi STM32](#)

## Guida passo passo: come utilizzare i driver LCD LVGL v9 con i dispositivi STM32

### Introduzione

Questa guida intende fornire istruzioni dettagliate su come configurare l'HAL STM32Cube con i nuovi driver per display TFT-LCD introdotti in LVGL v9.0. Il codice di esempio è stato testato sulla scheda Nucleo-F746ZG basata su STM32F746 con un pannello LCD basato su ST7789 connesso tramite SPI. L'applicazione stessa e il codice di configurazione hardware sono stati generati con il tool STM32CubeIDE 1.14.0.

### 💡 Suggerimento

ST Micro fornisce i propri driver TFT-LCD nel pacchetto di estensione software X-CUBE-DISPLAY. Sebbene questi driver possano essere utilizzati anche con LVGL, i driver LCD LVGL non dipendono da questo pacchetto.

I driver LCD LVGL sono concepiti come un'API alternativa e semplice da usare per implementare il supporto LCD per i progetti basati su LVGL su qualsiasi piattaforma. Inoltre, anche nella versione iniziale supportiamo più controller LCD di quanti ne fornisca attualmente X-CUBE-DISPLAY e prevediamo di aggiungere il supporto per un numero ancora maggiore di controller LCD in futuro.

Si noti tuttavia che, a differenza di X-CUBE-DISPLAY, i driver LCD LVGL non implementano la parte di comunicazione, che sia SPI, bus parallelo i2C o altro. È responsabilità dell'utente implementarli e ottimizzarli sulla piattaforma scelta. LVGL fornirà solo esempi per le piattaforme più diffuse.

Seguendo i passaggi descritti, si otterrà un programma completamente funzionante, utilizzabile come base per un progetto basato su LVGL. Se si ha fretta e non si è interessati ai dettagli, si può trovare il progetto finale [qui](#). Si dovrà solo configurare LVGL per utilizzare il driver corrispondente al proprio hardware (se diverso dall'ST7789) e implementare la funzione `ui_init()` per creare i propri widget.

### i Nota

Questo esempio non intende rappresentare la migliore implementazione possibile, né la soluzione consigliata. Si basa esclusivamente sui driver HAL forniti da ST Micro, che privilegiano la portabilità rispetto alle prestazioni. Ciò nonostante, le prestazioni sono molto buone, grazie all'efficiente implementazione dei driver basata su DMA.

### i Nota

Sebbene l'esempio utilizzi FreeRTOS, questo non è un requisito obbligatorio per i driver dei display LCD LVGL.

È possibile trovare frammenti di codice sorgente di questa guida nell'esempio `lv_port_lcd_stm32_template.c`.

## Configurazione hardware

In questo esempio utilizzeremo la periferica SPI1 per collegare il microcontrollore al pannello LCD. Oltre ai pin SPI controllati dall'hardware SCK e MOSI, abbiamo bisogno di alcuni pin di uscita aggiuntivi per la selezione del chip, la selezione dei comandi/dati e il reset dell'LCD:

pin	configurazione	LCD	label utente
PA4	GPIO_Output	CS	LCD_CS
PA5	SPI1_SCK	SCK	--
PA7	SPI1_MOSI	SDI	--
PA15	GPIO_Output	RESET	LCD_RESET
PB10	GPIO_Output	DC	LCD_DCX

## Istruzioni passo passo

1. Creare un nuovo progetto in File/New/STM32.
2. Selezionare il processore/scheda target.
3. Impostare il nome e la posizione del progetto.
4. Impostare il "Targeted Project Type" su STM32Cube e premere Finish.
5. Rispondi "Yes" per inizializzare le periferiche col loro "Mode" di default? Dopo la creazione del progetto, il file di configurazione (.ioc) si apre automaticamente.
6. Passare alla scheda "Pinout & Configuration".
7. Nella categoria "System Core" passare a RCC.
8. Impostare "High Speed Clock" a "BYPASS Clock Source" e "Low Speed Clock" a "Crystal/Ceramic Resonator".
9. Nella categoria "System Core" selezionare SYS e impostare "Timebase Source" su un valore diverso da SysTick (nel nostro esempio, TIM2).
10. Passare alla ""Clock Configuration".
11. Impostare la frequenza di clock HCLK al valore massimo (216 MHz per STM32F746).
12. Tornare alla scheda "Pinout & Configuration" e, nella categoria "Middleware and Software Packs", selezionare FREERTOS.
13. Selezionare "Interface: CMSIS\_V1".
14. Nella scheda "Advanced Settings" abilitare USE\_NEolib\_REENTRANT. Abbiamo terminato.
15. Nella vista "Pinout" configurare PA5 su SPI1\_SCK, PA7 su SPI1\_MOSI (clic destro del mouse sul pin e si seleziona la funzione).
16. Nella categoria "Pinout & Configuration/Connectivity" selezionare SPI1.
17. Impostare "Mode" su "Transmit Only Master" e "Hardware NSS Signal" a Disable.
18. Nella sottofinestra "Configuration" passare a "Parameter Settings".
19. Impostare "Frame Format" su Motorola, "Data Size" su "8 Bits", "First Bit" su "MSB First".
20. Impostare il Prescaler al valore massimo in base alla scheda tecnica del controller LCD (ad esempio, 15 MBit/s). Impostare "CPOL/CPHA" come richiesto (lasciare i valori di default).
21. Impostare "NSSP Mode" su Disabled e "NSS Signal Type" su Software.
22. In "DMA Settings" aggiungere una nuova "Request" per SPI1\_TX (quando si usa SPI1).
23. Impostare "Priority" su Medium, "Data Width" su "Half Word".
24. In "NVIC Settings" abilitare l'interrupt globale SPI1.

25. In "GPIO Settings" impostare "SPI1\_SCK" su "Pull-down" e "Very High output speed" e impostare "User Label" su `LCD_SCK`.
26. impostare "SPI1\_MOSI" su Pull-up e Very High, e rinominarlo `LCD_SDI`.
27. Selezionare la categoria "System Core/GPIO". Nella vista Pinout configurare pin aggiuntivi per la selezione del chip, il reset e la selezione dei comandi/dati. Chiamarli rispettivamente `LCD_CS`, `LCD_RESET` e `LCD_DCX`. Configurarli come "GPIO Output". (In questo esempio useremo PA4 per `LCD_CS`, PA15 per `LCD_RESET` e PB10 per `LCD_DCX`.)
28. Impostare `LCD_CS` su "No pull-up" e "no pull-down", "Low level" e "Very High speed".
29. Impostare `LCD_RESET` to "Pull-up" e "High level".
30. Impostare `LCD_DCX` su "No pull-up" e "no pull-down", "High level" e "Very High speed".
31. Aprire la scheda "Project Manager" e selezionare "Advanced Settings". Sul lato destro si trova la finestra "Register Callback". Selezionare SPI e impostarlo su ENABLE.
32. La configurazione hardware è pronta. Salvare la configurazione e lasciare che STM32Cube generi il codice sorgente.
33. Nell'alberatura del progetto, clonare il repository LVGL nella cartella "Middlewares/Third\_Party" (questo tutorial utilizza il branch "release/v9.0" di LVGL):

```
git clone https://github.com/lvgl/lvgl.git -b release/v9.0
```

34. La clonazione dovrebbe creare una sottocartella 'lvgl' all'interno della cartella 'Third\_Party'. Dalla cartella 'lvgl' copiare 'lv\_conf\_template.h' nella cartella 'Middlewares' e rinominarlo in 'lv\_conf.h'. Aggiorna l'alberatura del progetto.
35. Aprire 'lv\_conf.h' e alla riga 15 cambiare `#if 0` in `#if 1`.
36. Cercare la stringa `LV_USE_ST7735` e abilitare il driver LCD appropriato impostandone il valore su 1. Questo esempio utilizza il driver ST7789:

```
#define LV_USE_ST7789 1
```

37. Click destro sulla cartella 'Middlewares/Third\_Party/lvgl/tests', selezionare "Resource Configurations/Exclude from Build...", selezionare sia Debug che Release, quindi premere OK.
38. Click destro sul nome del progetto e selezionare "Properties". Nel pannello "C/C++ Build/Settings" selezionare MCU GCC Compiler/Include paths. Nel menu a discesa Configuration selezionare "[ All configurations ]". Aggiungere il seguente "Include path":

```
.../Middlewares/Third_Party/lvgl
```

39. Aprire Core/Src/stm32xxx\_it.c (il nome del file dipende dalla variante del processore). Aggiungere 'lv\_tick.h' alla sezione "Private includes":

```
/* Private includes -----*/
/* USER CODE BEGIN Includes */
#include "./src/tick/lv_tick.h"
/* USER CODE END Includes */
```

40. Trovare la funzione `TIM2_IRQHandler`. Aggiungere una chiamata a `lv_tick_inc()`:

```
void TIM2_IRQHandler(void)
{
    /* USER CODE BEGIN TIM2_IRQHandler 0 */

    /* USER CODE END TIM2_IRQHandler 0 */
    HAL_TIM_IRQHandler(&htim2);
    /* USER CODE BEGIN TIM2_IRQHandler 1 */
    lv_tick_inc(1);
    /* USER CODE END TIM2_IRQHandler 1 */
}
```

41. Salvare il file, quindi aprire "Core/Src/main.c". Aggiungere le seguenti righe a "Private includes" (se il vostro LCD utilizza un driver diverso da ST7789, sostituite il path e l'header del driver con quelli appropriati):

```
/* Private includes -----*/
/* USER CODE BEGIN Includes */
#include "lvgl.h"
#include "./src/drivers/display/st7789/lv_st7789.h"
/* USER CODE END Includes */
```

42. Aggiungere le seguenti righe alle definizioni private (modificarle in base alle specifiche del vostro LCD):

```
#define LCD_H_RES      240
#define LCD_V_RES      320
#define BUS_SPI1_POLL_TIMEOUT 0x1000U
```

43. Aggiungere le seguenti righe a "Private variables":

```
osThreadId LvglTaskHandle;
lv_display_t *lcd_disp;
volatile int lcd_bus_busy = 0;
```

44. Aggiungere la seguente riga a "Private function prototypes":

```
void ui_init(lv_display_t *disp);
void LVGL_Task(void const *argument);
```

45. Aggiungere le seguenti righe dopo USER CODE BEGIN RTOS\_THREADS:

```
osThreadDef(LvglTask, LVGL_Task, osPriorityIdle, 0, 1024);
LvglTaskHandle = osThreadCreate(osThread(LvglTask), NULL);
```

46. Copiare e incollare le funzioni di inizializzazione hardware e di callback di trasferimento dal codice di esempio dopo USER CODE BEGIN 4:

```
/* USER CODE BEGIN 4 */

void lcd_color_transfer_ready_cb(SPI_HandleTypeDef *hspi)
{
    /* CS high */
    HAL_GPIO_WritePin(LCD_CS_GPIO_Port, LCD_CS_Pin, GPIO_PIN_SET);
    lcd_bus_busy = 0;
    lv_display_flush_ready(lcd_disp);
}

/* Initialize LCD I/O bus, reset LCD */
static int32_t lcd_io_init(void)
{
    /* Register SPI Tx Complete Callback */
    HAL_SPI_RegisterCallback(&hspi, HAL_SPI_TX_COMPLETE_CB_ID, lcd_color_transfer_ready_cb);

    /* reset LCD */
    HAL_GPIO_WritePin(LCD_RESET_GPIO_Port, LCD_RESET_Pin, GPIO_PIN_RESET);
    HAL_Delay(100);
    HAL_GPIO_WritePin(LCD_RESET_GPIO_Port, LCD_RESET_Pin, GPIO_PIN_SET);
    HAL_Delay(100);

    HAL_GPIO_WritePin(LCD_CS_GPIO_Port, LCD_CS_Pin, GPIO_PIN_SET);
    HAL_GPIO_WritePin(LCD_DCX_GPIO_Port, LCD_DCX_Pin, GPIO_PIN_SET);

    return HAL_OK;
}

/* Platform-specific implementation of the LCD send command function. In general this should use polling transfer. */
static void lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    LV_UNUSED(disp);
    while (lcd_bus_busy); /* wait until previous transfer is finished */
    /* Set the SPI in 8-bit mode */
    hspi.Init.DataSize = SPI_DATASIZE_8BIT;
    HAL_SPI_Init(&hspi);
    /* DCX low (command) */
    HAL_GPIO_WritePin(LCD_DCX_GPIO_Port, LCD_DCX_Pin, GPIO_PIN_RESET);
    /* CS low */
    HAL_GPIO_WritePin(LCD_CS_GPIO_Port, LCD_CS_Pin, GPIO_PIN_RESET);
    /* send command */
    if (HAL_SPI_Transmit(&hspi, cmd, cmd_size, BUS_SPI1_POLL_TIMEOUT) == HAL_OK) {
        /* DCX high (data) */
        HAL_GPIO_WritePin(LCD_DCX_GPIO_Port, LCD_DCX_Pin, GPIO_PIN_SET);
        /* for short data blocks we use polling transfer */
        HAL_SPI_Transmit(&hspi, (uint8_t *)param, (uint16_t)param_size, BUS_SPI1_POLL_TIMEOUT);
        /* CS high */
        HAL_GPIO_WritePin(LCD_CS_GPIO_Port, LCD_CS_Pin, GPIO_PIN_SET);
    }
}

/* Platform-specific implementation of the LCD send color function. For better performance this should use DMA transfer.
 * In case of a DMA transfer a callback must be installed to notify LVGL about the end of the transfer.
 */
static void lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    LV_UNUSED(disp);
    while (lcd_bus_busy); /* wait until previous transfer is finished */
    /* Set the SPI in 8-bit mode */
    hspi.Init.DataSize = SPI_DATASIZE_8BIT;
    HAL_SPI_Init(&hspi);
    /* DCX low (command) */
    HAL_GPIO_WritePin(LCD_DCX_GPIO_Port, LCD_DCX_Pin, GPIO_PIN_RESET);
    /* CS low */
    HAL_GPIO_WritePin(LCD_CS_GPIO_Port, LCD_CS_Pin, GPIO_PIN_RESET);
    /* send command */
    if (HAL_SPI_Transmit(&hspi, cmd, cmd_size, BUS_SPI1_POLL_TIMEOUT) == HAL_OK) {
        /* DCX high (data) */
        HAL_GPIO_WritePin(LCD_DCX_GPIO_Port, LCD_DCX_Pin, GPIO_PIN_SET);
        /* for color data use DMA transfer */
        /* Set the SPI in 16-bit mode to match endianness */
        hspi.Init.DataSize = SPI_DATASIZE_16BIT;
        HAL_SPI_Init(&hspi);
        lcd_bus_busy = 1;
        HAL_SPI_Transmit_DMA(&hspi, param, (uint16_t)param_size / 2);
        /* NOTE: CS will be reset in the transfer ready callback */
    }
}
```

47. Aggiungere la funzione LVGL\_Task(). Sostituire la chiamata `lv_st7789_create()` con il driver appropriato. È possibile modificare l'orientamento di default modificando il parametro di `lv_display_set_rotation()`. Sarà inoltre necessario creare i "display buffer". Questo esempio utilizza uno schema di doppio buffering con buffer parziali di dimensioni pari a 1/10. Nella maggior parte dei casi questo è un buon compromesso tra la dimensione della memoria richiesta e le prestazioni, ma si può sperimentare con altre impostazioni.

```
void LVGL_Task(void const *argument)
{
    /* Initialize LVGL */
    lv_init();

    /* Initialize LCD I/O */
    if (lcd_io_init() != 0)
        return;

    /* Create the LVGL display object and the LCD display driver */
    lcd_disp = lv_st7789_create(LCD_H_RES, LCD_V_RES, LV_LCD_FLAG_NONE, lcd_send_cmd, lcd_send_color);
    lv_display_set_rotation(lcd_disp, LV_DISPLAY_ROTATION_270);

    /* Allocate draw buffers on the heap. In this example we use two partial buffers of 1/10th size of the screen */
    lv_color_t * buf1 = NULL;
    lv_color_t * buf2 = NULL;

    uint32_t buf_size = LCD_H_RES * LCD_V_RES / 10 * lv_color_format_get_size(lv_display_get_color_format(lcd_disp));

    buf1 = lv_malloc(buf_size);
    if(buf1 == NULL) {
        LV_LOG_ERROR("display draw buffer malloc failed");
        return;
    }

    buf2 = lv_malloc(buf_size);
    if(buf2 == NULL) {
        LV_LOG_ERROR("display buffer malloc failed");
        lv_free(buf1);
        return;
    }
    lv_display_set_buffers(lcd_disp, buf1, buf2, buf_size, LV_DISPLAY_RENDER_MODE_PARTIAL);

    ui_init(lcd_disp);

    for(;;) {
        /* The task running lv_timer_handler should have lower priority than that running `lv_tick_inc` */
        lv_timer_handler();
        /* raise the task priority of LVGL and/or reduce the handler period can improve the performance */
        osDelay(10);
    }
}
```

48. Non resta che implementare `ui_init()` per creare lo screen. Ecco un semplice esempio di "Hello World":

```
void ui_init(lv_display_t *disp)
{
    lv_obj_t *obj;

    /* set screen background to white */
    lv_obj_t *scr = lv_screen_active();
    lv_obj_set_style_bg_color(scr, lv_color_white(), 0);
    lv_obj_set_style_bg_opa(scr, LV_OPA_100, 0);

    /* create label */
    obj = lv_label_create(scr);
    lv_obj_set_align(widget, LV_ALIGN_CENTER);
    lv_obj_set_height(widget, LV_SIZE_CONTENT);
    lv_obj_set_width(widget, LV_SIZE_CONTENT);
    lv_obj_set_style_text_font(widget, &lv_font_montserrat_14, 0);
    lv_obj_set_style_text_color(widget, lv_color_black(), 0);
    lv_label_set_text(widget, "Hello World!");
}
```

## NXP eLCDIF

### Panoramica

eLCDIF è una periferica presente su alcuni dispositivi NXP in grado di pilotare pannelli di display tramite l'interfaccia RGB, supporta diverse profondità di colore e, sui dispositivi compatibili con MIPI-DSI, il suo output può essere indirizzato all'interfaccia fisica del display MIPI. Il driver NXP eLCDIF di LVGL è responsabile del "bind" [collegamento] del driver di basso livello dell'SDK MCUX di NXP al sottosistema di visualizzazione LVGL.

### Prerequisiti

- Questo driver si basa sulla presenza dell'SDK MCUX di NXP nello stesso progetto.
- Attivare il driver impostando `LV_USE_NXP_ELCDIF` a 1 nel file "`lv_conf.h`".

## Utilizzo

Il driver LVGL per eLCDIF presuppone che la piattaforma abbia già configurato il driver di basso livello del display, impostato il pin-mux, i clock, ecc. Richiede inoltre che l'indirizzo base della periferica e la struttura di configurazione siano già impostati.

Il codice seguente illustra l'utilizzo del driver in modalità ***LV\_DISPLAY\_RENDER\_MODE\_DIRECT***. Si noti che in questa modalità di funzionamento l'applicazione è responsabile dell'allocazione dello spazio dei framebuffer e del loro passaggio al display. Nell'esempio seguente **buffer1** e **buffer2** sono il buffer corrente e quello successivo che verranno copiati sullo schermo, scambiati a ogni operazione di flush (gestita internamente dal driver del display). Si noti inoltre che, in "direct mode", ogni buffer dovrebbe avere spazio sufficiente per contenere almeno le dimensioni dello schermo, ovvero l'altezza per la larghezza per i byte di un pixel (dipendente dall'applicazione o supportato dal display). Nel codice seguente, questa dimensione è rappresentata da **buf\_size**.

```
elcdif_rgb_mode_config_t config;
ELCDIF_RgbModeGetDefaultConfig(&config);

lv_display_t * g_disp = lv_nxp_display_elcdif_create_direct(LCDIF, config, buffer1, buffer2, buf_size);
lv_display_set_default(g_disp);
```

Per utilizzare il driver in modalità ***LV\_DISPLAY\_RENDER\_MODE\_PARTIAL***, è necessario allocare un buffer aggiuntivo, preferibilmente nella regione di memoria più veloce disponibile.

Lo swapping del buffer può essere attivato passando un secondo buffer della stessa dimensione al posto dell'argomento **NULL**. Si noti che in questo caso **BUF\_SIZE** deve avere almeno uno spazio sufficiente a contenere dati pari a 1/10 delle dimensioni effettive del display.

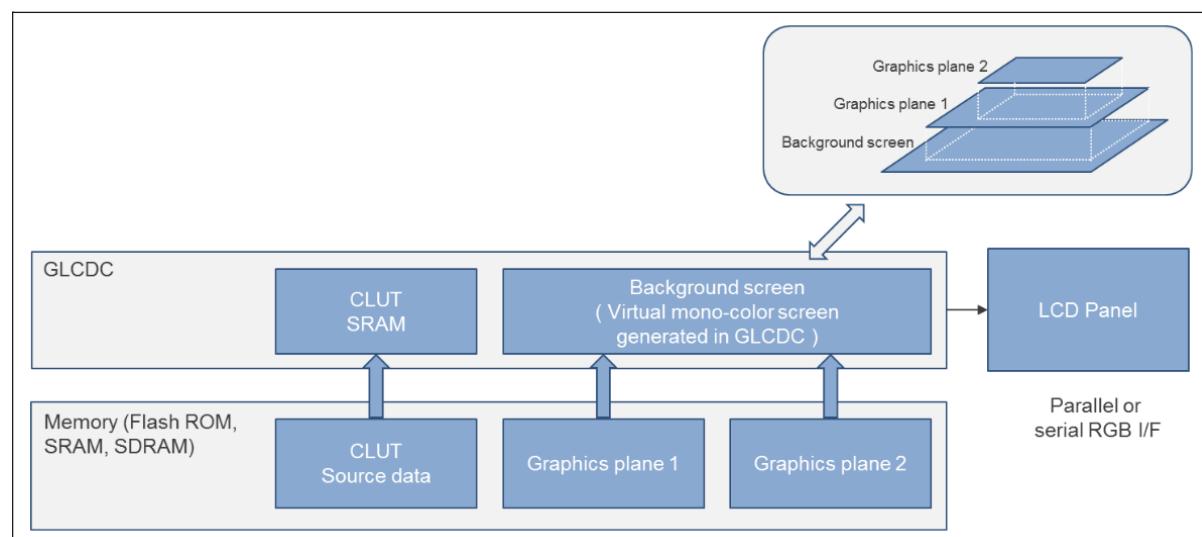
```
#define BUF_SIZE (DISPLAY_HEIGHT * DISPLAY_WIDTH / 10 * 2) /*1/10 screen size for RGB565 format*/
static uint8_t partial_draw_buf[BUF_SIZE];
lv_display_t * g_disp = lv_nxp_display_elcdif_create_partial(LCDIF, config, partial_draw_buf, NULL, BUF_SIZE);
```

In fase di runtime, la funzione di gestione eventi del driver eLCDIF deve essere chiamata all'interno del gestore di interrupt eLCDIF. Questa funzione è responsabile della notifica al sottosistema di visualizzazione LVGL del completamento dell'operazione di flush:

```
void eLCDIF_IRQHandler(void)
{
    lv_nxp_display_elcdif_event_handler(g_disp);
}
```

## Renesas GLCDC

### Panoramica



<br/>

GLCDC è una periferica di output grafico multistadio utilizzata nei microcontrollori Renesas. È progettata per generare automaticamente segnali di temporizzazione e dati per diversi pannelli LCD.

- Supporta pannelli LCD con interfaccia RGB (fino a 24 bit) e segnali di sincronizzazione (HSYNC, VSYNC e Data Enable opzionali)
- Supporta vari formati colore per i piani grafici in ingresso (RGB888, ARGB8888, RGB565, ARGB1555, ARGB4444, CLUT8, CLUT4, CLUT1)
- Supporta l'utilizzo della Color Look-Up Table (CLUT) per i piani grafici in ingresso (ARGB8888) con 512 word (32 bit/word)
- Supporta vari formati colore per l'output (RGB888, RGB666, RGB565, Serial RGB888)
- Può inserire due piani grafici sopra il piano di background e fonderli sullo schermo
- Genera un dot clock per il pannello. La sorgente di clock è selezionabile tra interna o esterna (LCD\_EXTCLK).
- Supporta la regolazione della luminosità, del contrasto e la correzione gamma.
- Supporta gli interrupt GLCDC per gestire la commutazione del frame buffer o il rilevamento dell'underflow.

La configurazione di un progetto e l'ulteriore integrazione con l'ecosistema Renesas sono descritte in dettaglio nella [pagina Renesas](#). Consultare i seguenti repository per esempi pronti all'uso:

- [EK-RA8D1](#)
- [EK-RA6M3G](#)
- [RX72N Envision Kit](#)

## Prerequisiti

- Questo diver si basa sul codice generato da e<sup>2</sup> studio. Omettere questo passaggio durante la configurazione del progetto causerà un errore di compilazione.
- Activate the diver by setting `LV_USE_RENESAS_GLCDC` to 1 in your `lv_conf.h`.

## Utilizzo

Non è necessario implementare funzioni specifiche per la piattaforma.

Il codice seguente illustra l'utilizzo del diver in modalità `LV_DISPLAY_RENDER_MODE_DIRECT`.

```
lv_display_t * disp = lv_renesas_glcdfc_direct_create();
lv_display_set_default(disp);
```

Per utilizzare il driver in modalità `LV_DISPLAY_RENDER_MODE_PARTIAL`, è necessario allocare un buffer aggiuntivo, preferibilmente nella regione di memoria più veloce disponibile.

Lo swapping del buffer può essere attivato passando un secondo buffer della stessa dimensione al posto dell'argomento `NULL`.

```
static lv_color_t partial_draw_buf[DISPLAY_HSIZE_INPUT0 * DISPLAY_VSIZE_INPUT0 / 10] BSP_PLACE_IN_SECTION(".sram") BSP_ALIGN_VARIABLE(1024);
lv_display_t * disp = lv_renesas_glcdfc_partial_create(partial_draw_buf, NULL, sizeof(partial_draw_buf));
lv_display_set_default(disp);
```

### Nota

La modalità parziale può essere attivata tramite la macro nel file `src/board_init.c` dei progetti demo.

## Rotazione dello schermo

La rotazione dello schermo basata su software è supportata in modalità parziale. Utilizza l'API comune, non è richiesta alcuna configurazione aggiuntiva:

```
lv_display_set_rotation(lv_display_get_default(), LV_DISP_ROTATION_90);
/* OR */
lv_display_set_rotation(lv_display_get_default(), LV_DISP_ROTATION_180);
/* OR */
lv_display_set_rotation(lv_display_get_default(), LV_DISP_ROTATION_270);
```

Assicurarsi che l'heap sia sufficientemente grande, poiché verrà allocato un buffer con le stesse dimensioni del buffer parziale.

## Driver LTDC STM32

Alcuni STM32 dispongono di una periferica specializzata per la gestione dei display chiamata LTDC (LCD-TFT display controller).

### Modalità di utilizzo con LVGL

Il driver all'interno di LVGL è progettato per funzionare con una periferica LTDC già configurata. Si basa sull'HAL per rilevare le informazioni sulla configurazione. Il formato colore del display LVGL creato corrisponderà al formato colore del layer LTDC. Utilizzare STM32CubeIDE o STM32CubeMX per generare il codice di inizializzazione LTDC.

Esistono diversi casi d'uso per il driver LVGL. Tutte le permutazioni delle opzioni seguenti sono ben supportate.

- buffer singolo o doppio
- modalità di rendering diretta o parziale
- Con sistema operativo e senza sistema operativo
- flushing parallelo con DMA2D (solo per la modalità di rendering parziale)

Se il sistema operativo è abilitato, verrà utilizzata una primitiva di sincronizzazione per dare al thread la possibilità di cedere il passo ad altri thread mentre è bloccato, migliorando l'utilizzo della CPU. Vedere [LV\\_USE\\_OS](#) in your lv\_conf.h

## Layer LTDC

Questo driver crea un display LVGL che si occupa solo di uno specifico layer della periferica LTDC, il che significa che è possibile creare due display LVGL LTDC che operano indipendentemente sui rispettivi layer.

### Direct Render Mode

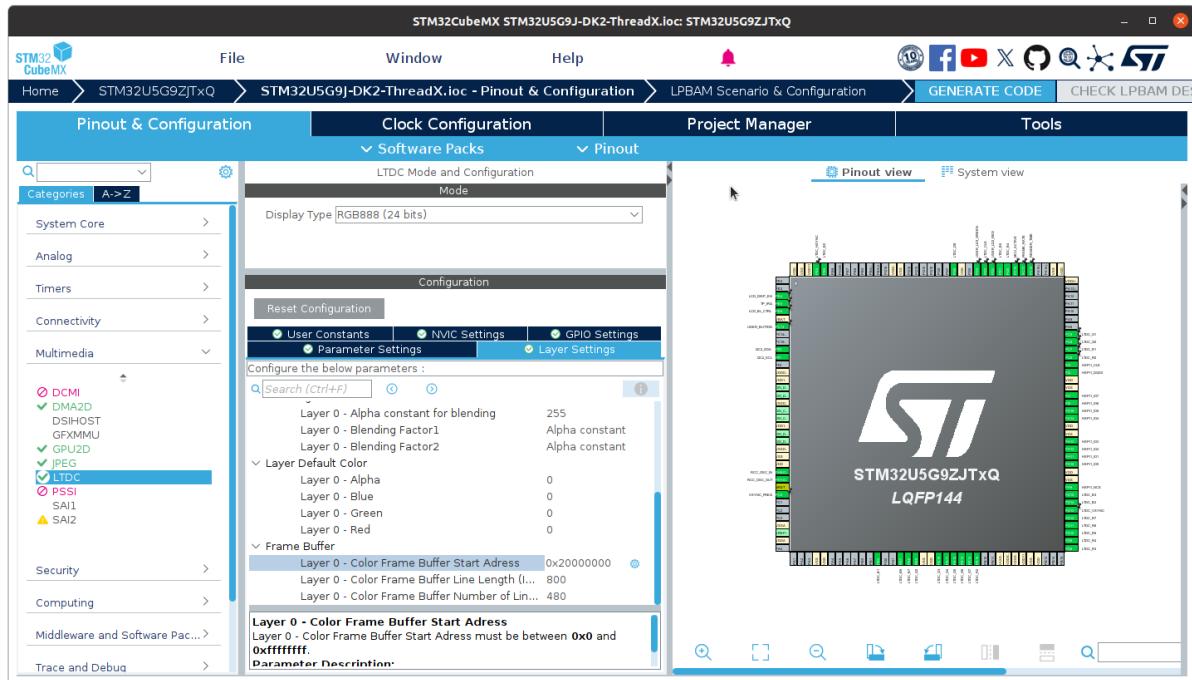
Per la modalità di rendering diretto, richiamare `lv_st_ltdc_create_direct()` in questo modo:

```
void * my_ltdc_framebuffer_address = (void *)0x20000000u;
uint32_t my_ltdc_layer_index = 0; /* typically 0 or 1 */
lv_display_t * disp = lv_st_ltdc_create_direct(my_ltdc_framebuffer_address,
                                             optional_other_full_size_buffer,
                                             my_ltdc_layer_index);
```

`my_ltdc_framebuffer_address` è il framebuffer configurato per l'utilizzo da parte di LTDC. `optional_other_full_size_buffer` può essere un altro buffer delle stesse dimensioni del framebuffer di default per la modalità a doppio buffer, oppure `NULL` in caso contrario. `my_ltdc_layer_index` è l'indice del layer LTDC per cui creare il display.

Per ottenere i migliori risultati visivi, si consiglia di utilizzare `optional_other_full_size_buffer` se la memoria disponibile è sufficiente. La modalità a buffer singolo è quella da utilizzare se la memoria è molto scarsa. Se la memoria disponibile è quasi sufficiente per la modalità diretta a doppio buffer, ma non del tutto, utilizzare la modalità di rendering parziale.

Per chiarire cosa sia esattamente `my_ltdc_framebuffer_address`, si tratta del valore di `pLayerCfg.FBStartAddress` quando il layer LTDC viene configurato utilizzando l'HAL STM32, che viene scritto nel registro CFBAR della periferica del layer LTDC.



## Partial Render Mode

Per la modalità di rendering parziale, invocare `lv_st_ltdc_create_partial()` in questo modo:

```
static uint8_t partial_buf1[65536];
static uint8_t optional_partial_buf2[65536];
uint32_t my_ltdc_layer_index = 0; /* typically 0 or 1 */
lv_display_t * disp = lv_st_ltdc_create_partial(partial_buf1,
                                                optional_partial_buf2,
                                                65536,
                                                my_ltdc_layer_index);
```

Il driver utilizzerà le informazioni nella configurazione del layer LTDC per trovare il framebuffer del layer e scaricarlo.

Fornire un secondo buffer parziale può migliorare l'utilizzo della CPU e aumentare le prestazioni rispetto a un singolo buffer se `LV_ST_LTDC_USE_DMA2D_FLUSH` è abilitato.

## Script del Linker

È necessario assicurarsi che la memoria del framebuffer LTDC sia effettivamente riservata nello script del linker. Questo è un file che normalmente ha l'estensione `.ld`. Nell'esempio seguente, `1125K` è specificato perché la profondità di colore è 24 (3 byte per pixel), la larghezza del display è 800, l'altezza del display è 480 e `1K` significa 1024 byte.  $3 \times 800 \times 480 \div 1024 = 1125$ . La somma delle voci RAM (`FB_RAM + RAM`) dev'essere uguale alla RAM totale del dispositivo.

```
/* Memories definition */
MEMORY
{
    FB_RAM  (xrw)   : ORIGIN = 0x20000000, LENGTH = 1125K /* single 24bit 800x480 buffer */
    RAM     (xrw)   : ORIGIN = 0x20119400, LENGTH = 1883K
    FLASH   (rx)    : ORIGIN = 0x08000000, LENGTH = 4096K
}
```

## Rotazione del Display

Il driver supporta la rotazione del display con `lv_display_set_rotation(disp, rotation)` dove `rotation` è una tra `LV_DISP_ROTATION_90`, `LV_DISP_ROTATION_180` o `LV_DISP_ROTATION_270`. La rotazione è inizialmente `LV_DISP_ROTATION_0`.

La rotazione viene eseguita tramite software e funziona solo se il display è stato creato utilizzando `lv_st_ltdc_create_partial()`. `LV_ST_LTDC_USE_DMA2D_FLUSH` non avrà alcun effetto se si utilizza la rotazione.

## DMA2D

`LV_ST_LTDC_USE_DMA2D_FLUSH` può essere abilitato per utilizzare DMA2D per svuotare i buffer parziali in parallelo con altre attività LVGL, indipendentemente dal fatto che il sistema operativo sia abilitato o meno. Se la visualizzazione non è parziale, non è necessario abilitare questa opzione.

Inoltre, è possibile combinare layer con formato colore `LV_COLOR_FORMAT_ARGB1555` sopra i layer `LV_COLOR_FORMAT_RGB565` utilizzando DMA2D.

Non deve essere abilitato contemporaneamente a `LV_USE_DRAW_DMA2D`. Vedere [DMA2D support](#).

### Ulteriori Informazioni

Si potrebbe essere interessati ad abilitare il [Nema GFX renderer](#) se l'STM32 ha una GPU NeoChrom.

`lv_port_riverdi_stm32u5` è un modo rapido per iniziare a usare LTDC su LVGL.

## Driver del controller LCD ST7735

### Panoramica

`ST7735S` è un controller/driver a chip singolo per LCD TFT-LCD grafico a 262K colori. È costituito da 396 circuiti di pilotaggio per linee sorgente e 162 per linee di gate. Questo chip è in grado di connettersi direttamente a un microprocessore esterno e accetta interfacce SPI (Serial Peripheral Interface) e interfacce parallele a 8/9/16/18 bit. I dati del display possono essere memorizzati nella RAM integrata del display da 132 x 162 x 18 bit. Può eseguire operazioni di lettura/scrittura dei dati di visualizzazione nella RAM senza bisogno di un clock esterno, riducendo al minimo il consumo energetico. Inoltre, grazie ai circuiti di alimentazione integrati necessari per pilotare i cristalli liquidi, è possibile realizzare un sistema di visualizzazione con un numero inferiore di componenti.

Il `driver` del controller LCD ST7735, è un driver "platform-agnostic" [indipendente dalla piattaforma], basato sul `driver generico MIPI`. Implementa l'inizializzazione del display, supporta la rotazione del display e implementa il callback di flush del display. L'utente deve implementare solo due funzioni specifiche della piattaforma per inviare un comando o dati pixel al controller tramite SPI o bus parallelo. In genere, queste vengono implementate richiamando le funzioni della libreria SDK appropriate sulla piattaforma specificata.

### Prerequisiti

Non ci sono prerequisiti.

### Configurazione del driver

Abilitare il supporto del driver ST7735 in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig

```
#define LV_USE_ST7735 1
```

### Utilizzo

È necessario implementare due funzioni dipendenti dalla piattaforma:

```
/* Send short command to the LCD. This function shall wait until the transaction finishes. */
int32_t my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    ...
}

/* Send large array of pixel data to the LCD. If necessary, this function has to do the byte-swapping. This function can do the transfer in the background. */
int32_t my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    ...
}
```

Per creare un display basato su ST7735 utilizzare la funzione

```
/*
 * Create an LCD display with ST7735 driver
 * @param hor_res      horizontal resolution
 * @param ver_res      vertical resolution
 * @param flags        default configuration settings (mirror, RGB ordering, etc.)
 * @param send_cmd     platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
 * @param send_color   platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' ↴
 * @param callback     pointer to the created display
 */
lv_display_t * lv_st7735_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                lv_st7735_send_cmd_cb_t send_cmd_cb, lv_st7735_send_color_cb_t send_color_cb);
```

Per ulteriori dettagli e un esempio pratico, consultare la documentazione generica del driver MIPI.

### Nota

È possibile trovare una guida passo passo e l'implementazione effettiva dei callback su un STM32F746 utilizzando STM32CubeIDE e le librerie ST HAL qui: *Guida passo passo: come utilizzare i driver LCD LVGL v9 con i dispositivi STM32*

## Driver per Controller LCD ST7789

### Panoramica

L'**ST7789** è un controller/driver a chip singolo per TFT-LCD grafico a 262K colori. È costituito da 720 circuiti di pilotaggio per linee "source" e 320 per linee di gate. Questo chip è in grado di connettersi direttamente a un microprocessore esterno e accetta interfacce parallele a 8/9/16/18 bit. I dati del display possono essere memorizzati nella RAM integrata da 240x320x18 bit. Può eseguire operazioni di lettura/scrittura dei dati di visualizzazione nella RAM senza bisogno di un clock esterno, riducendo al minimo il consumo energetico. Inoltre, grazie al circuito di alimentazione integrato necessario per pilotare i cristalli liquidi, è possibile realizzare un sistema di visualizzazione con il minor numero di componenti.

Il **driver** del controller LCD ST7789 è un driver indipendente dalla piattaforma, basato sul **driver generico MIPI**. Implementa l'inizializzazione del display, supporta la rotazione del display e implementa il callback di flush del display. L'utente deve implementare solo due funzioni specifiche della piattaforma per inviare un comando o dati pixel al controller tramite SPI o bus parallelo. In genere, queste vengono implementate richiamando le funzioni della libreria SDK appropriate sulla piattaforma specificata.

### Prerequisiti

Non ci sono prerequisiti.

### Configurazione del driver

Abilitare il supporto del driver ST7789 in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig

```
#define LV_USE_ST7789 1
```

### Utilizzo

È necessario implementare due funzioni dipendenti dalla piattaforma:

```
/* Send short command to the LCD. This function shall wait until the transaction finishes. */
int32_t my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    ...
}

/* Send large array of pixel data to the LCD. If necessary, this function has to do the byte-swapping. This function can do the transfer in the ↴
 * background. */
int32_t my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    ...
}
```

Per creare un display basato su ST7789 utilizzare la funzione

```
/*
 * Create an LCD display with ST7789 driver
 * @param hor_res      horizontal resolution
 * @param ver_res      vertical resolution
 * @param flags        default configuration settings (mirror, RGB ordering, etc.)
 * @param send_cmd     platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
 * @param send_color   platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' ↴
 * @param callback     pointer to the created display
 */
lv_display_t * lv_st7789_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                lv_st7789_send_cmd_cb_t send_cmd_cb, lv_st7789_send_color_cb_t send_color_cb);
```

Per ulteriori dettagli e un esempio pratico, consultare la documentazione generica del driver MIPI.

### Nota

È possibile trovare una guida passo passo e l'implementazione effettiva dei callback su un STM32F746 utilizzando STM32CubeIDE e le librerie ST HAL qui: *Guida passo passo: come utilizzare i driver LCD LVGL v9 con i dispositivi STM32*

## Driver per Controller LCD ST7796

### Panoramica

ST7796S è un controller/driver a chip singolo per display LCD TFT grafico a 262K colori. È costituito da 960 linee sorgente e 480 linee di gate per il pilotaggio dei circuiti. ST7796S è in grado di connettersi direttamente a un microprocessore esterno e accetta interfacce parallele a 8/9/16/18 bit, SPI; inoltre, fornisce anche un'interfaccia MIPI. I dati del display possono essere memorizzati nella RAM integrata da 320x480x18 bit. Può eseguire operazioni di lettura/scrittura della RAM dei dati del display senza clock esterno, riducendo al minimo il consumo energetico. Inoltre, grazie al circuito di alimentazione integrato necessario per pilotare i cristalli liquidi, è possibile realizzare un sistema di visualizzazione con un numero inferiore di componenti.

Il **driver** del controller LCD ST7796 è un driver indipendente dalla piattaforma, basato sul **rive** generico **MIP**. Implementa l'inizializzazione del display, supporta la rotazione del display e implementa il callback di flush del display. L'utente deve implementare solo due funzioni specifiche della piattaforma per inviare un comando o dati pixel al controller tramite SPI o bus parallelo. In genere, queste vengono implementate richiamando le funzioni della libreria SDK appropriate sulla piattaforma specificata.

### Prerequisiti

Non ci sono prerequisiti.

### Configurazione del driver

Abilitare il supporto del driver ST7796 in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig

```
#define LV_USE_ST7796 1
```

### Utilizzo

È necessario implementare due funzioni dipendenti dalla piattaforma:

```
/* Send short command to the LCD. This function shall wait until the transaction finishes. */
int32_t my_lcd_send_cmd(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)
{
    ...
}

/* Send large array of pixel data to the LCD. If necessary, this function has to do the byte-swapping. This function can do the transfer in the ↴
 * background. */
int32_t my_lcd_send_color(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)
{
    ...
}
```

Per creare un display basato su ST7796 utilizzare la funzione

```
/*
 * Create an LCD display with ST7796 driver
 * @param hor_res      horizontal resolution
 * @param ver_res      vertical resolution
 * @param flags        default configuration settings (mirror, RGB ordering, etc.)
 * @param send_cmd     platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
 * @param send_color   platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' ↴
 * @param callback     pointer to the created display
 */
lv_display_t * lv_st7796_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                lv_st7796_send_cmd_cb_t send_cmd_cb, lv_st7796_send_color_cb_t send_color_cb);
```

For additional details and a working example see the [generic MIPI driver documentation](#).

### Nota

È possibile trovare una guida passo passo e l'implementazione effettiva dei callback su un STM32F746 utilizzando STM32CubeIDE e le librerie ST HAL qui: [Guida passo passo: come utilizzare i driver LCD LVGL v9 con i dispositivi STM32](#)

## 4.5.2 Driver Libinput

### Panoramica

Libinput è uno stack di input per processi che devono fornire eventi da dispositivi di input di uso comune. Tra questi rientrano mouse, tastiere, touchpad, touchscreen e tavolette grafiche. Libinput gestisce le peculiarità specifiche del dispositivo e fornisce un'API di facile utilizzo per ricevere eventi dai dispositivi.

### Prerequisiti

È installata la versione di sviluppo di libinput (di solito `libinput-dev`). Se il dispositivo di input richiede peculiarità, assicurarsi che siano installate (di solito in `/usr/share/libinput/*.quirks`). Per verificare se il dispositivo è configurato correttamente per l'utilizzo con libinput, è possibile eseguire `libinput list-devices`.

```
$ sudo libinput list-devices
...
Device:      ETPS/2 Elantech Touchpad
Kernel:      /dev/input/event5
Group:       10
Seat:        seat0, default
Size:        102x74mm
Capabilities: pointer gesture
Tap-to-click: disabled
Tap-and-drag: enabled
...
```

Se il dispositivo non viene visualizzato, potrebbe essere necessario configurare udev e le regole udev appropriate per connetterlo.

Inoltre, per il supporto completo della tastiera, incluse lettere e modificatori, è necessario installare la versione di sviluppo di libxkbcommon (solitamente `libxkbcommon-dev`).

### Configurazione del driver

Abilitare il supporto del driver libinput in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig.

```
#define LV_USE_LIBINPUT 1
```

Il supporto completo della tastiera deve essere abilitato separatamente.

```
#define LV_LIBINPUT_XKB      1
#define LV_LIBINPUT_XKB_KEY_MAP { .rules = NULL, .model = "pc101", .layout = "us", .variant = NULL, .options = NULL }
```

Per trovare i valori corretti per la mappatura dei tasti, è possibile utilizzare il comando `setxkbmap -query`.

## Utilizzo

Per impostare un dispositivo di input tramite il driver libinput, è sufficiente chiamare `lv_libinput_create` on il rispettivo tipo di dispositivo (LV\_INDEV\_TYPE\_POINTER o LV\_INDEV\_TYPE\_KEYPAD) e il path del nodo del dispositivo (ad esempio /dev/input/event5).

```
lv_indev_t *indev = lv_libinput_create(LV_INDEV_TYPE_POINTER, "/dev/input/event5");
```

Si noti che i touchscreen vengono trattati come dispositivi di puntamento (assoluti) dal driver libinput e richiedono LV\_INDEV\_TYPE\_POINTER.

A seconda del sistema, i path dei nodi dei dispositivi potrebbero non essere stabili dopo i riavvii. In questo caso, è possibile utilizzare `lv_libinput_find_dev` per trovare il primo dispositivo con una funzionalità specifica.

```
char *path = lv_libinput_find_dev(LV_LIBINPUT_CAPABILITY_TOUCH, true);
```

Il secondo argomento controlla se tutti i dispositivi vengono nuovamente scansionati o meno. Se si hanno molti dispositivi collegati, l'operazione può risultare piuttosto lenta. Pertanto, è consigliabile specificare `true` solo alla prima chiamata quando si richiama questo metodo più volte di seguito. Per trovare tutti i dispositivi con una funzionalità specifica, utilizzare `lv_libinput_find_devs`.

Per collegare una tastiera a un'area di testo, creare un gruppo di input dedicato e impostarlo sia sull'indev che sulla textarea.

```
lv_obj_t *textarea = lv_textarea_create(...);
...
lv_group_t *keyboard_input_group = lv_group_create();
lv_indev_set_group(indev, keyboard_input_group);
lv_group_add_obj(keyboard_input_group, textarea);
```

## 4.5.3 Driver Display/Input OpenGL ES

### Panoramica

The OpenGL ES display/input driver offers support for simulating the LVGL display and keyboard/mouse inputs in an desktop window created via GLFW.

È un'alternativa a Wayland, XCB, SDL o Qt.

Lo scopo principale di questo driver è testare/eseguire il debug dell'applicazione LVGL in una finestra di simulazione OpenGL.

### Prerequisiti

Il driver OpenGL utilizza GLEW GLFW per accedere al gestore finestre OpenGL.

1. Installare GLEW e GLFW: `sudo apt-get install libglew-dev libglfw3-dev`

### Configurare il driver OpenGL

1. Librerie linkate richieste: -IGL -IGLEW -lglfw
2. Abilitare il supporto del driver OpenGL in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig

```
#define LV_USE_OPENGL 1
```

### Utilizzo di Base

```
#include "lvgl/lvgl.h"
#include "lvgl/examples/lv_examples.h"
#include "lvgl/demos/lv_demos.h"

#define WIDTH 640
#define HEIGHT 480

int main()
{
    /* initialize lvgl */
    lv_init();

    /* create a window and initialize OpenGL */
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_glfw_window_t * window = lv_glfw_window_create(WIDTH, HEIGHT, true);

/* create a display that flushes to a texture */
lv_display_t * texture = lv_opengles_texture_create(WIDTH, HEIGHT);
lv_display_set_default(texture);

/* add the texture to the window */
unsigned int texture_id = lv_opengles_texture_get_texture_id(texture);
lv_glfw_texture_t * window_texture = lv_glfw_window_add_texture(window, texture_id, WIDTH, HEIGHT);

/* get the mouse indev of the window texture */
lv_indev_t * mouse = lv_glfw_texture_get_mouse_indev(window_texture);

/* add a cursor to the mouse indev */
LV_IMAGE_DECLARE(mouse_cursor_icon);
lv_obj_t * cursor_obj = lv_image_create(lv_screen_active());
lv_image_set_src(cursor_obj, &mouse_cursor_icon);
lv_indev_set_cursor(mouse, cursor_obj);

/* create Widgets on the screen */
lv_demo_widgets();

while (1)
{
    uint32_t time_until_next = lv_timer_handler();
    if(time_until_next == LV_NO_TIMER_READY) time_until_next = LV_DEF_REFR_PERIOD; /*handle LV_NO_TIMER_READY. Another option is to `sleep` for longer*/
    lv_delay_ms(time_until_next);
}

return 0;
}

```

## Utilizzo Avanzato

Il driver OpenGL può disegnare le texture dall'utente. È possibile utilizzare una libreria di terze parti per aggiungere contenuto a una texture e il driver disegnerà la texture nella finestra.

```

#include "lvgl/lvgl.h"
#include <GL/glew.h>
#include <GLFW/glfw3.h>

#define WIDTH 640
#define HEIGHT 480

void custom_texture_example(void)
{
    /*****
     * MAIN WINDOW
     *****/

    /* create window and initialize OpenGL */
    /* multiple windows can be created */
    lv_glfw_window_t * window = lv_glfw_window_create(WIDTH, HEIGHT, true);

    /*****
     * OPTIONAL MAIN TEXTURE
     *****/

    /* create main display that flushes to a texture */
    lv_display_t * main_texture = lv_opengles_texture_create(WIDTH, HEIGHT);
    lv_display_set_default(main_texture);

    /* add the main texture to the window */
    unsigned int main_texture_id = lv_opengles_texture_get_texture_id(main_texture);
    lv_glfw_texture_t * window_main_texture = lv_glfw_window_add_texture(window, main_texture_id, WIDTH, HEIGHT);

    /* get the mouse indev of this main texture */
    lv_indev_t * main_texture_mouse = lv_glfw_texture_get_mouse_indev(window_main_texture);

    /* add a cursor to the mouse indev */
    LV_IMAGE_DECLARE(mouse_cursor_icon);
    lv_obj_t * cursor_obj = lv_image_create(lv_screen_active());
    lv_image_set_src(cursor_obj, &mouse_cursor_icon);
    lv_indev_set_cursor(main_texture_mouse, cursor_obj);

    /* create Widgets on the screen of the main texture */
    lv_demo_widgets();

    /*****
     * ANOTHER TEXTURE
     *****/

    /* create a sub display that flushes to a texture */
    const int32_t sub_texture_w = 300;
    const int32_t sub_texture_h = 300;
    lv_display_t * sub_texture = lv_opengles_texture_create(sub_texture_w, sub_texture_h);

    /* add the sub texture to the window */
    unsigned int sub_texture_id = lv_opengles_texture_get_texture_id(sub_texture);
    lv_glfw_texture_t * window_sub_texture = lv_glfw_window_add_texture(window, sub_texture_id, sub_texture_w, sub_texture_h);

    /* create Widgets on the screen of the sub texture */
    lv_display_set_default(sub_texture);
    lv_example_keyboard_2();
    lv_display_set_default(main_texture);

    /* position the sub texture within the window */
    lv_glfw_texture_set_x(window_sub_texture, 250);
    lv_glfw_texture_set_y(window_sub_texture, 150);

    /* optionally change the opacity of the sub texture */
    lv_glfw_texture_set_opa(window_sub_texture, LV_OPA_80);

    /*****
     * USE AN EXTERNAL OPENGL TEXTURE IN LVGL
     *****/

```

(continues on next page)

(continua dalla pagina precedente)

```
*****
unsigned int external_texture_id;
 glGenTextures(1, &external_texture_id);
 glBindTexture(GL_TEXTURE_2D, external_texture_id);
 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MIN_FILTER, GL_LINEAR);
 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_MAG_FILTER, GL_LINEAR);
 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_S, GL_CLAMP_TO_EDGE);
 glTexParameteri(GL_TEXTURE_2D, GL_TEXTURE_WRAP_T, GL_CLAMP_TO_EDGE);
 glPixelStorei(GL_UNPACK_ALIGNMENT, 1);
 LV_IMAGE_DECLARE(img_cogwheel_argb);

#if LV_COLOR_DEPTH == 8
 const int texture_format = GL_R8;
#elif LV_COLOR_DEPTH == 16
 const int texture_format = GL_RGB565;
#elif LV_COLOR_DEPTH == 24
 const int texture_format = GL_RGB;
#elif LV_COLOR_DEPTH == 32
 const int texture_format = GL_RGBA;
#else
#error("Unsupported color format")
#endif
 glTexImage2D(GL_TEXTURE_2D, 0, texture_format, img_cogwheel_argb.header.w, img_cogwheel_argb.header.h, 0, GL_BGRA, GL_UNSIGNED_BYTE, img_cogwheel_argb.data);
 glGenerateMipmap(GL_TEXTURE_2D);
 glBindTexture(GL_TEXTURE_2D, 0);

 /* add the external texture to the window */
 lv_glfw_texture_t * window_external_texture = lv_glfw_window_add_texture(window, external_texture_id, img_cogwheel_argb.header.w, img_cogwheel_argb.header.h);

 /* set the position and opacity of the external texture within the window */
 lv_glfw_texture_set_x(window_external_texture, 20);
 lv_glfw_texture_set_y(window_external_texture, 20);
 lv_glfw_texture_set_opa(window_external_texture, LV_OPA_70);

 *****
 * USE AN LVGL TEXTURE IN ANOTHER LIBRARY
*****
```

lv\_refr\_now(sub\_texture);  
/\* the texture is drawn on by LVGL and can be used by anything that uses OpenGL textures \*/  
third\_party\_lib\_use\_texture(sub\_texture\_id);  
}

## Renderer con Cache delle Texture OpenGL

In LVGL è presente un renderer che memorizza nella cache le aree renderizzate tramite software come texture OpenGL. Le texture vengono recuperate dalla cache e riutilizzate quando viene rilevata una corrispondenza. Nella maggior parte dei casi, le prestazioni saranno notevolmente migliorate.

```
#define LV_USE_DRAW_OPENGL 1
```

## Limitazioni Note

- Le prestazioni saranno le stesse o leggermente peggiori se le aree disegnate non vengono mai trovate nella cache a causa di widget con colori o forme che variano continuamente. Un esempio è un'etichetta il cui colore è impostato su un valore casuale per ogni fotogramma, come nella scena "Multiple labels" della demo del benchmark.
- I layer con pixel trasparenti e una trasparenza complessiva del layer non si fonderanno correttamente. L'effetto può essere osservato nella scena "Containers with opa\_layer" della demo del benchmark negli angoli del bordo.
- I layer con rotazione non sono attualmente supportati. Le immagini con rotazione funzionano correttamente.

## 4.5.4 Touchpad

### Driver Linux Evdev

#### Panoramica

"Event device" (evdev) di Linux, è un'API indipendente dall'hardware che consente l'accesso agli eventi di input, ad esempio da un mouse o da un touchscreen. Viene esposto tramite l'interfaccia del file system del device Linux.

#### Prerequisiti

Il sistema dispone di un dispositivo di input configurato (solitamente in /dev/input/, ad esempio /dev/input/event0).

## Configurazione del driver

Abilitare il supporto del driver evdev LVGL Linux in `lv_conf.h`.

```
#define LV_USE_EVDEV 1
```

## Utilizzo

Per impostare un input di evento, creare prima un "input device" con `lv_eudev_create` impostandolo sul dispositivo di eventi Linux corretto. Quindi collegarlo al display LVGL con `lv_indev_set_display`.

```
lv_indev_t *touch = lv_eudev_create(LV_INDEV_TYPE_POINTER, "/dev/input/event0");
lv_indev_set_display(touch, disp);
```

Assicurarsi che un oggetto `lv_display_t` sia già stato creato per `disp`. Un esempio è mostrato di seguito, utilizzando il driver framebuffer Linux.

```
lv_display_t * disp = lv_linux_fbdev
lv_linux_fbdev_set_file(disp, "/dev/fb0");_create();
```

## Individuazione del dispositivo di input

Se non si riesce a determinare il dispositivo di input, eseguire prima

```
`$cat /proc/bus/input/devices`
```

Questo dovrebbe mostrare i dispositivi di input e ci saranno voci con la parola `event` che forniscono un indizio sul dispositivo da utilizzare, ad esempio `event1` sarebbe `/dev/input/event1`.

Si può usare `evtest` per visualizzare i dati da quella sorgente di eventi e verificare se è effettivamente quella desiderata.

Provare:

```
$evtest /dev/input/event1 sostituendo eventX con il dispositivo di eventi indicato sopra.
```

## Rilevamento automatico dei dispositivi di input

È disponibile il supporto per la ricerca e l'aggiunta automatica di dispositivi di input in `/dev/input/`. I nuovi dispositivi verranno aggiunti automaticamente quando vengono connessi. Per abilitare questa funzionalità, è sufficiente chiamare `lv_eudev_discovery_start(NULL, NULL)`.

Si potrebbe voler reagire all'aggiunta di un nuovo dispositivo in modo che possa essere applicata un'immagine del cursore, ad esempio. È possibile fornire una funzione di callback che verrà richiamata quando viene aggiunto un nuovo dispositivo.

```
#include "lvgl/src/core/lv_global.h"

static void indev_deleted_cb(lv_event_t * e)
{
    if(LV_GLOBAL_DEFAULT()->deinit_in_progress) return;
    lv_obj_t * cursor_obj = lv_event_get_user_data(e);
    lv_obj_delete(cursor_obj);
}

static void discovery_cb(lv_indev_t * indev, lv_eudev_type_t type, void * user_data)
{
    LV_LOG_USER("new '%s' device discovered", type == LV_EDEV_TYPE_REL ? "REL" :
               type == LV_EDEV_TYPE_ABS ? "ABS" :
               type == LV_EDEV_TYPE_KEY ? "KEY" :
               "unknown");

    if(type == LV_EDEV_TYPE_REL) {
        LV_IMAGE_DECLARE(mouse_cursor_icon);
        lv_obj_t * cursor_obj = lv_image_create(lv_screen_active());
        lv_image_set_src(cursor_obj, &mouse_cursor_icon);
        lv_indev_set_cursor(indev, cursor_obj);
        lv_indev_add_event_cb(indev, indev_deleted_cb, LV_EVENT_DELETE, cursor_obj);
    }
}

int main()
{
    /* ... */
    lv_eudev_discovery_start(discovery_cb, NULL);
    /* ... */
}
```

Al momento della stesura di questo documento, questa funzionalità non è supportata in BSD.

## FT6X36

TODO [da fare]

### 4.5.5 Driver per Display/Input Wayland

#### Panoramica

Il **driver Wayland** offre supporto per la simulazione del display LVGL e degli input da tastiera/mouse in una finestra del desktop.

È un'alternativa a **X11** o a **SDL2**

The main purpose for this driver is for testing/debugging the LVGL application. It can also be used to run applications in "kiosk mode".

#### Dipendenze

Il driver Wayland richiede alcune dipendenze.

Su Ubuntu

```
sudo apt-get install libwayland-dev libxkbcommon-dev libwayland-bin wayland-protocols
```

Su Fedora

```
sudo dnf install wayland-devel libxkbcommon-devel wayland-utils wayland-protocols-devel
```

#### Configurazione del driver Wayland

1. Abilitare il driver Wayland in `lv_conf.h`

```
#define LV_USE_WAYLAND 1
```

2. Opzioni di configurazione facoltative:

Alcune impostazioni opzionali dipendono dall'abilitazione del supporto del buffer DMA (`LV_WAYLAND_USE_DMABUF`). La tabella seguente riassume le combinazioni valide e le relative limitazioni:

Tabella 2: Possibilità di configurazione

Opzione di configurazione	Senza DMABUF	Con DMABUF
<code>LV_DRAW_USE_G2D</code>	Non richiesto	<b>Obbligatorio</b>
<code>LV_WAYLAND_BUF_COUNT</code>	1	1 o 2
<code>LV_WAYLAND_RENDER_MODE</code>	<code>LV_DISPLAY_RENDER_MODE_PARTIAL</code>	<code>LV_DISPLAY_RENDER_MODE_DIRECT</code> o <code>LV_DISPLAY_RENDER_MODE_FULL</code>
<code>LV_WAYLAND_WINDOW_DECORATIONS</code>	1 o 0	0
<code>LV_WAYLAND_WL_SHELL</code> ( <b>Obsoleto</b> )	Facoltativo (1 o 0)	Facoltativo (1 o 0)

Note aggiuntive

- Il supporto DMABUF (`LV_WAYLAND_USE_DMABUF`) migliora le prestazioni e abilita più modalità di rendering, ma presenta requisiti e restrizioni specifici.
- `LV_WAYLAND_WINDOW_DECORATIONS` è richiesto solo per alcuni compositori (ad esempio, GNOME/Mutter o Weston).
- `LV_WAYLAND_WL_SHELL` Abilita il supporto per `wl_shell` ormai obsoleto. Utile solo quando il BSP sul target ha weston 9.x. (**Obsoleto** e verrà rimosso in una versione futura)

## Esempio

Un simulatore di esempio è disponibile in questo [repo](#)

## Utilizzo

1. In `main.c` `#include "lv_drivers/wayland/wayland.h"`
2. Abilitare il driver Wayland in `lv_conf.h` con `LV_USE_WAYLAND 1`
3. `LV_COLOR_DEPTH` dovrebbe essere impostato su 32 o su 16 in `lv_conf.h`
4. Aggiungere un display utilizzando `lv_wayland_window_create()`, possibilmente con una callback di chiusura per tracciare lo stato di ciascun display:

```
#define H_RES (800)
#define V_RES (480)

/* Create a display */
lv_disp_t * disp = lv_wayland_create_window(H_RES, V_RES, "Window Title", close_cb);
```

Come parte della chiamata precedente, il driver Wayland registrerà quattro dispositivi di input per ciascun display:

- un KEYPAD connesso agli eventi della tastiera Wayland
- un POINTER connesso agli eventi touch Wayland
- un POINTER connesso agli eventi del puntatore Wayland
- un ENCODER connesso agli assi del puntatore Wayland

Gli handle per i dispositivi di input di ciascun display possono essere ottenuti utilizzando rispettivamente `lv_wayland_get_indev_keyboard()`, `lv_wayland_get_indev_touchscreen()`, `lv_wayland_get_indev_pointer()` e `lv_wayland_get_indev_pointeraxis()`.

## Modalità fullscreen

Per impostare la finestra a schermo intero tramite programmazione, utilizzare la funzione `lv_wayland_window_set_fullscreen()` rispettivamente con `true` o `false` per l'argomento `fullscreen`.

## Modalità massimizzata

Per massimizzare la finestra tramite programmazione, utilizzare la funzione `lv_wayland_window_set_maximized()` rispettivamente con `true` o `false` per l'argomento `maximized`.

## Gestore timer personalizzato

Chiamare sempre `lv_wayland_timer_handler()` nel proprio loop del timer invece del normale `lv_timer_handler()`.

**Nota:** `lv_wayland_timer_handler()` chiama internamente `lv_timer_handler()`

Questo consente al client Wayland di funzionare correttamente su Weston; il ridimensionamento dei buffer di memoria condivisa durante un commit non funziona correttamente su Weston.

Il wrapping della chiamata a `lv_timer_handler()` è necessario per avere un maggiore controllo su quando viene chiamata la callback di flush LVGL.

## Building del driver Wayland

Un simulatore di esempio è disponibile in questo [repo](#)

Se è necessario utilizzare il driver con un altro sistema di build. I file sorgente e fli header per la shell XDG devono essere generati dalle definizioni per il protocollo della shell XDG.

Nell'esempio, CMake viene utilizzato per eseguire l'operazione richiamando l'utility `wayland-scanner`

Per eseguire questa operazione manualmente,

Assicurarsi che le dipendenze elencate all'inizio dell'articolo siano installate.

Il protocollo Wayland è definito utilizzando file XML presenti in `/usr/share/wayland-protocols`

Per generare i file richiesti, eseguire i seguenti comandi:

```
wayland-scanner client-header $SYSROOT/usr/share/wayland-protocols/stable/xdg-shell/xdg-shell.xml wayland_xdg_shell.h
wayland-scanner private-code $SYSROOT/usr/share/wayland-protocols/stable/xdg-shell/xdg-shell.xml wayland_xdg_shell.c
```

Quando `LV_WAYLAND_USE_DMABUF` è impostato a 1, devono essere generati anche i seguenti protocolli:

```
wayland-scanner client-header $SYSROOT/usr/share/wayland-protocols/stable/linux-dmabuf/linux-dmabuf-v1.xml wayland_linux_dmabuf.h
wayland-scanner private-code $SYSROOT/usr/share/wayland-protocols/stable/linux-dmabuf/linux-dmabuf-v1.xml wayland_linux_dmabuf.c
```

I file risultanti possono quindi essere integrati nel progetto. È consigliabile eseguire nuovamente `wayland-scanner` su ogni build per garantire che vengano generate le versioni corrette, che devono corrispondere alla versione della libreria a "dynamically linked" `wayland-client` installata sul sistema.

## Stato attuale e obiettivi

- Aggiungere la modalità di rendering diretto
- Refactoring delle integrazioni della shell per evitare un'eccessiva compilazione condizionale
- Tecnicamente, il driver Wayland consente di creare più finestre, ma questa funzionalità è sperimentale.
- Aggiungere in futuro un supporto avanzato per la shell XDG per consentire la creazione di applicazioni desktop su piattaforme Unix-like, in modo simile a quanto avviene con il driver Win32.
- Aggiungere il supporto per Mesa; attualmente viene utilizzato `wl_shm`, ma non è la tecnica più efficace.

## Segnalazioni di bug

Il driver Wayland è attualmente in fase di sviluppo, segnalazioni di bug, contributi e feedback sono sempre benvenuti.

È tuttavia importante creare problemi dettagliati quando si riscontra un problema; log e screenshot del problema sono di grande aiuto.

Abilitare `LV_USE_LOG` e avviare l'eseguibile del simulatore in questo modo.

```
WAYLAND_DEBUG=1 ./path/to/simulator_executable > /tmp/debug 2>&1
```

Questo creerà un file di log chiamato `debug` nella directory `/tmp`, eseguire il copia e incolla del contenuto del file nell'issue su GitHub. Il file di log contiene i log LVGL e i messaggi Wayland.

Assicurarsi di replicare il problema rapidamente, altrimenti i log diventano troppo grandi.

## 4.5.6 Driver Display/Input di Windows

### Panoramica

Il `driver display/Input di Windows` offre supporto per la simulazione del display LVGL e degli input da tastiera/mouse in una finestra Win32 di Windows.

Lo scopo principale di questo driver è testare/debuggere l'applicazione LVGL in una finestra di simulazione di **Windows** tramite la **simulator mode**, o sviluppare un'applicazione desktop standard di **Windows** con LVGL tramite la **application mode**.

Queste sono le **somiglianze** tra la modalità simulatore e la modalità applicazione.

- Supporto per l'integrazione di dispositivi LVGL per puntatore, tastiera e encoder.
- Supporto per l'input touch di Windows.
- Supporto per l'integrazione del metodo di input di Windows.
- Supporto per il riconoscimento DPI per monitor (sia V1 che V2).
- Fornire interoperabilità basata su HWND per altre infrastrutture dell'interfaccia utente di Windows.

Queste sono le **differenze** tra la modalità simulatore e la modalità applicazione.

## Modalità Simulatore

- Progettata per lo scenario di simulazione di dispositivi LVGL: simula il rendering LVGL su un pannello di display hardware.
- Mantiene costante la risoluzione del display LVGL per simulare al meglio il layout dell'interfaccia utente che verrà visualizzato sui dispositivi di produzione.
- Quando si modifica l'impostazione di ridimensionamento DPI di Windows, il backend di Windows estenderà il contenuto del display.

## Modalità Applicazione

- Progettata per lo scenario di sviluppo di applicazioni desktop Windows.
- Supporta il ridimensionamento delle finestre e la risoluzione del display LVGL viene modificata dinamicamente.
- Quando si modifica l'impostazione di ridimensionamento DPI di Windows, viene modificato anche il valore DPI del display LVGL.
- La risoluzione impostata per lv\_windows\_create\_display corrisponde alla dimensione della finestra anziché alla dimensione del client della finestra, per seguire le convenzioni di altre infrastrutture di interfaccia utente desktop di Windows.
- Le applicazioni basate su questa modalità dovrebbero adattare la modifica della risoluzione del display LVGL per supportare correttamente il ridimensionamento delle finestre.

## Prerequisiti

Il requisito minimo del sistema operativo Windows per questo driver è Windows Vista RTM.

Se si utilizzano librerie `shim` API Windows come [YY-Thunks](#), il requisito minimo del sistema operativo Windows testato per questo driver è Windows XP RTM.

In base all'API GDI di Windows utilizzata da questo driver, è possibile che il requisito minimo del sistema operativo Windows per questo driver sia Windows 2000 RTM.

## Configurare il Driver Windows

Abilitare il supporto del driver Windows in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig

```
#define LV_USE_WINDOWS 1
```

## Utilizzo

```
#include <Windows.h>
#include "lvgl/lvgl.h"
#include "lvgl/examples/lv_examples.h"
#include "lvgl/demos/lv_demos.h"

int main()
{
    lv_init();

    int32_t zoom_level = 100;
    bool allow_dpi_override = false;
    bool simulator_mode = false;
    lv_display_t* display = lv_windows_create_display(
        L"LVGL Display Window",
        800,
        480,
        zoom_level,
        allow_dpi_override,
        simulator_mode);
    if (!display)
    {
        return -1;
    }

    lv_lock();

    lv_indev_t* pointer_device = lv_windows_acquire_pointer_indev(display);
    if (!pointer_device)
    {
        return -1;
    }
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_indev_t* keypad_device = lv_windows_acquire_keypad_indev(display);
{
    return -1;
}

lv_indev_t* encoder_device = lv_windows_acquire_encoder_indev(display);
{
    return -1;
}

lv_demo_widgets();
lv_unlock();

while (1)
{
    uint32_t time_till_next = lv_timer_handler();
    if(time_till_next == LV_NO_TIMER_READY) time_till_next = LV_DEF_REFR_PERIOD; /*handle LV_NO_TIMER_READY. Another option is to 'sleep' for longer*/
    lv_delay_ms(time_till_next);
}

return 0;
}

```

## 4.5.7 Driver Display/Input X11

### Panoramica

Il **driver Display/Input X11** offre supporto per la simulazione del display LVGL e degli input da tastiera/mouse in una finestra del desktop X11.

È un'alternativa a **Wayland, XCB, SDL o Qt**.

Lo scopo principale di questo driver è testare/eseguire il debug dell'applicazione LVGL in una finestra di simulazione **Linux**.

### Prerequisiti

Il driver X11 utilizza XLib per accedere al gestore delle finestre Linux.

1. Installare XLib: `sudo apt-get install libx11-6` (dovrebbe essere già installato)
2. Installare il pacchetto di sviluppo XLib: `sudo apt-get install libx11-dev`

### Configurare il driver X11

1. Abilitare il supporto del driver X11 in `lv_conf.h`, tramite la definizione del compilatore `cmake` o tramite KConfig

```
#define LV_USE_X11 1
```

2. Opzioni di configurazione facoltative:

- Direct Exit

```
#define LV_X11_DIRECT_EXIT 1 /* preferred default - ends the application automatically if last window has been closed */
// or
#define LV_X11_DIRECT_EXIT 0 /* application is responsible for ending the application (e.g. by own LV_EVENT_DELETE*/
// handler */
```

- Double buffering

```
#define LV_X11_DOUBLE_BUFFER 1 /* preferred default */
// or
#define LV_X11_DOUBLE_BUFFER 0 /* not recommended */
```

- Render mode

```
#define LV_X11_RENDER_MODE_PARTIAL 1 /* LV_DISPLAY_RENDER_MODE_PARTIAL, preferred default */
// or
#define LV_X11_RENDER_MODE_DIRECT 1 /* LV_DISPLAY_RENDER_MODE_DIRECT, not recommended for X11 driver */
// or
#define LV_X11_RENDER_MODE_DULL 1 /* LV_DISPLAY_RENDER_MODE_FULL, not recommended for X11 driver */
```

## Utilizzo

Inizializzazione minima che apre una finestra e abilita il supporto per tastiera/mouse  
(ad esempio, in main.c, LV\_X11\_DIRECT\_EXIT dev'essere 1):

```
int main(int argc, char ** argv)
{
    ...

    /* initialize X11 display driver */
    lv_display_t * disp = lv_x11_window_create("LVGL X11 Simulation", monitor_hor_res, monitor_ver_res);

    /* initialize X11 input drivers (for keyboard, mouse & mousewheel) */
    lv_x11_inputs_create(disp, NULL);

    ...

    while(true)
    {
        ...

        /* Periodically call the lv_timer handler */
        lv_timer_handler();
    }
}
```

Inizializzazione completa con simbolo del puntatore del mouse e gestione dell'uscita dall'applicazione  
(dipendente da LV\_X11\_DIRECT\_EXIT (può essere 1 o 0))

```
bool terminated = false;

#if !LV_X11_DIRECT_EXIT
static void on_close_cb(lv_event_t * e)
{
    ...

    terminate = true;
}
#endif

int main(int argc, char ** argv)
{
    ...

    /* initialize X11 display driver */
    lv_display_t * disp = lv_x11_window_create("LVGL X11 Simulation", monitor_hor_res, monitor_ver_res);
    lv_display_add_event_cb(disp, on_close_cb, LV_EVENT_DELETE, disp);

    /* initialize X11 input drivers (for keyboard, mouse & mousewheel) */
    LV_IMAGE_DECLARE(my_mouse_cursor_icon);
    lv_x11_inputs_create(disp, &my_mouse_cursor_icon);

    #if !LV_X11_DIRECT_EXIT
    /* set optional window close callback to enable application cleanup and exit */
    lv_x11_window_set_close_cb(disp, on_close_cb, disp);
    #endif

    ...

    while(!terminated)
    {
        ...

        /* Periodically call the lv_timer handler */
        lv_timer_handler();
    }
}
```

## 4.5.8 Driver per Display/Input UEFI

### Panoramica

Il **driver** display/input di **UEFI** offre supporto per l'utilizzo di LVGL con UEFI.

### Prerequisiti

Sono necessari i seguenti protocolli disponibili:

- **EFI\_LOADED\_IMAGE\_PROTOCOL\_GUID**, per il supporto del file system (utilizzato per determinare il file system utilizzato per caricare l'applicazione)
- **EFI\_SIMPLE\_FILE\_SYSTEM\_PROTOCOL\_GUID**, per il supporto del file system
- **EFI\_SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL\_GUID**, per il supporto della tastiera

- *EFI\_SIMPLE\_POINTER\_PROTOCOL\_GUID*, per il supporto del mouse
- *EFI\_ABSOLUTE\_POINTER\_PROTOCOL\_GUID*, per il supporto touch
- *EFI\_GRAPHICS\_OUTPUT\_PROTOCOL\_GUID*, per il disegno
- *EFI\_EDID\_ACTIVE\_PROTOCOL\_GUID*, per il disegno (facoltativo)

## Configurare il driver UEFI

- Abilitare il supporto del driver UEFI in lv\_conf.h

```
#define LV_USE_UEFI 1
```

- Abilitare le funzioni core della memoria, che sono wrapper per AllocatePool e FreePool (utilizzando memoria di tipo *EfiBootServicesData*) se non si desidera utilizzare le proprie implementazioni.

```
#define LV_UEFI_USE_MEMORY_SERVICES 1
```

- È possibile abilitare il supporto del file system per il file system da cui è stata caricata l'applicazione (lettera di default 'E')

```
#define LV_USE_FS_UEFI 1
```

- È necessario definire un file "include" che contenga le definizioni UEFI di base (protocolli e tipi); sono disponibili 2 file predefiniti che possono essere utilizzati per EDK2 e gnu-efi.

```
#define LV_USE_UEFI_INCLUDE <lvgl/src/drivers/uefi/lv_uefi_edk2.h>
// or
#define LV_USE_UEFI_INCLUDE <lvgl/src/drivers/uefi/lv_uefi_gnu_efi.h>
```

## Utilizzo

```
#include "lvgl/lvgl.h"
#include "lvgl/examples/lv_examples.h"
#include "lvgl/demos/lv_demos.h"

EFI_STATUS EFIAPI EfiMain(EFI_HANDLE ImageHandle, EFI_SYSTEM_TABLE * SystemTable)
{
    lv_uefi_init(ImageHandle, SystemTable);
    lv_init();

    if(!lv_is_initialized()) return EFI_NOT_READY;

    EFI_HANDLE handle = NULL;
    lv_display_t* display = NULL;
    lv_indev_t* indev = NULL;
    lv_group_t* group = NULL;
    lv_obj_t* cursor = NULL;
    // used to get out of the main loop
    size_t counter;

    // Init the display
    handle = lv_uefi_display_get_active();
    if(handle == NULL) {
        handle = lv_uefi_display_get_any();
    }
    if(handle == NULL) {
        lv_deinit();
        return EFI_UNSUPPORTED;
    }

    display = lv_uefi_display_create(handle);
    lv_display_set_default(display);

    // Create the group
    group = lv_group_create();
    lv_group_set_default(group);

    // Create an image that can be used as cursor
    cursor = lv_image_create(lv_layer_top());
    lv_image_set_src(cursor, "E:cursor.png");

    // Create the input devices
    indev = lv_uefi_simple_text_input_indev_create();
    lv_indev_set_group(indev, lv_group_get_default());
    lv_uefi_simple_text_input_indev_add_all(indev);

    indev = lv_uefi_simple_pointer_indev_create(NULL);
    lv_uefi_simple_pointer_indev_add_all(indev);
    lv_indev_set_cursor(indev, cursor);

    indev = lv_uefi_absolute_pointer_indev_create(NULL);
    lv_uefi_absolute_pointer_indev_add_all(indev);

    lv_demo_widgets();

    // Run main loop for ~ 10 seconds
    counter = 0;
    while(counter < 10000) {
        counter++;
        gBS->Stall(1000);
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_tick_inc(1);
    lv_timer_handler();
}

return EFI_SUCCESS;
}

```

## 4.5.9 Driver SDL

### Panoramica

SDL (Simple DirectMedia Layer) offre un modo multipiattaforma per gestire grafica, input e contenuti multimediali, rendendolo una scelta eccellente per l'esecuzione di applicazioni LVGL su PC.

### Prerequisiti

Installare SDL in base alla piattaforma.

- Linux `sudo apt install libsdl2-dev`
- MacOS `brew install sdl2`
- Windows <https://github.com/libsdl-org/SDL/releases>

### Configurare il driver SDL

1. Librerie linkate richieste: -lSDL2
2. Abilita il supporto del driver SDL in `lv_conf.h`, nelle definizioni del compilatore CMake o in KConfig.

```

#define LV_USE SDL 1
#define LV SDL_INCLUDE_PATH <SDL2/SDL.h>
#define SDL_HOR_RES 400
#define SDL_VER_RES 400

```

### Utilizzo di Base

```

#include <unistd.h>
#define SDL_MAIN_HANDLED /*To fix SDL's "undefined reference to WinMain" issue*/
#include <SDL2/SDL.h>
#include "drivers/sdl/lv_sdl_mouse.h"
#include "drivers/sdl/lv_sdl_mousewheel.h"
#include "drivers/sdl/lv_sdl_keyboard.h"

static lv_display_t *lvDisplay;
static lv_indev_t *lvMouse;
static lv_indev_t *lvMouseWheel;
static lv_indev_t *lvKeyboard;

#if LV_USE_LOG != 0
static void lv_log_print_g_cb(lv_log_level_t level, const char * buf)
{
    LV_UNUSED(level);
    LV_UNUSED(buf);
}
#endif

int main()
{
    /* initialize lvgl */
    lv_init();

    // Workaround for sdl2 `~m32` crash
    // https://bugs.launchpad.net/ubuntu/+source/libSDL2/+bug/1775067/comments/7
#ifndef WIN32
    setenv("DBUS_FATAL_WARNINGS", "0", 1);
#endif

    /* Register the log print callback */
#if LV_USE_LOG != 0
    lv_log_register_print_cb(lv_log_print_g_cb);
#endif

    /* Add a display
     * Use the 'monitor' driver which creates window on PC's monitor to simulate a display*/
    lvDisplay = lv_sdl_window_create(SDL_HOR_RES, SDL_VER_RES);
    lvMouse = lv_sdl_mouse_create();
    lvMouseWheel = lv_sdl_mousewheel_create();
    lvKeyboard = lv_sdl_keyboard_create();

    /* create Widgets on the screen */
    lv_demo_widgets();

    Uint32 lastTick = SDL_GetTicks();
    while(1) {
        SDL_Delay(5);
        Uint32 current = SDL_GetTicks();

```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_tick_inc(current - lastTick); // Update the tick timer. Tick is new for LVGL 9
    lastTick = current;
    lv_timer_handler(); // Update the UI
}

return 0;
}

```

## Utilizzo di un IDE

LVGL con SDL è stato portato su vari IDE.

- **Eclipse con driver SDL:** Consigliato su Linux e Mac, supporta anche CMake
- **VSCode con driver SDL:** Consigliato su Linux (SDL) e Mac (SDL)
- **Linux Generico:** Progetto basato su CMake in cui è possibile passare facilmente da fbdev a DRM e SDL.

## 4.6 Renderer e GPU

### 4.6.1 Renderer Software

#### API

```

lv_draw_sw_blend.h
lv_draw_sw_blend_private.h
lv_draw_sw_blend_to_al88.h
lv_draw_sw_blend_to_argb8888.h
lv_draw_sw_blend_to_argb8888_premultiplied.h
lv_draw_sw_blend_to_i1.h
lv_draw_sw_blend_to_l8.h
lv_draw_sw_blend_to_rgb565.h
lv_draw_sw_blend_to_rgb565_swapped.h
lv_draw_sw_blend_to_rgb888.h
lv_draw_sw.h
lv_draw_sw_grad.h
lv_draw_sw_mask.h
lv_draw_sw_mask_private.h
lv_draw_sw_private.h
lv_draw_sw_utils.h

```

### 4.6.2 GPU Arm-2D

Arm-2D non è una GPU, ma **un livello di astrazione per GPU 2D dedicato ai microcontrollori**. Supporta tutti i processori Cortex-M, dal Cortex-M0 al più recente Cortex-M85.

Arm-2D accelera LVGL9 con due modalità: **Modalità Sincrona** e **Modalità Asincrona**.

- Quando sono disponibili **Helium** e **ACI (Arm Custom Instruction)**, si consiglia di utilizzare la **Modalità Sincrona** per accelerare LVGL.
- Quando sono disponibili GPU 2D basate su Arm-2D, ad esempio **GPU 2D basate su DMAC-350**, si consiglia di utilizzare la **Modalità Asincrona** per accelerare LVGL.

Arm-2D è un progetto open source su GitHub. Per ulteriori informazioni, consultare: <https://github.com/ARM-software/Arm-2D>.

## Come si Usa

In generale:

- è possibile impostare la macro `LV_USE_DRAW_ARM2D_SYNC` a 1 e `LV_DRAW_SW_ASM` a `LV_DRAW_SW_ASM_HELIUM` in `lv_conf.h` per abilitare l'accelerazione sincrona Arm-2D per LVGL.
- È possibile impostare la macro `LV_USE_DRAW_ARM2D_ASYNC` a 1 in `lv_conf.h` per abilitare l'accelerazione asincrona Arm-2D per LVGL.

Se si utilizza CMSIS-Pack per il deploy di LVGL. Non è necessario definire manualmente la macro `LV_USE_DRAW_ARM2D_SYNC`, ma `lv_conf_cmsis.h` controllerà l'ambiente e imposterà `LV_USE_DRAW_ARM2D_SYNC` di conseguenza.

## Considerazioni Progettuali

Come accennato in precedenza, Arm-2D è un layer di astrazione per GPU 2D; pertanto, se non è disponibile un acceleratore o un set di istruzioni dedicato (come Helium o ACI) per Arm-2D, l'aumento delle prestazioni per LVGL è trascurabile (a volte addirittura peggiore) per i normali processori Cortex-M.

**Consigliamo vivamente di abilitare l'accelerazione Arm-2D per LVGL quando:**

- I processori target sono **Cortex-M55, Cortex-M52 e Cortex-M85**
- I processori target supportano **Helium**.
- Il fornitore del dispositivo fornisce un driver compatibile con ARM-2D per i propri acceleratori 2D proprietari e/o ACI (Arm Customized Instruction).
- Il dispositivo target contiene **DMAC-350**

## Esempi

- Un progetto MDK basato su Cortex-M55 (supporta Helium), disponibile per l'emulazione PC.

## API

`lv_draw_sw_arm2d.h`

`lv_blend_arm2d.h`

### 4.6.3 Acceleratore di Elaborazione Pixel Espressif

## API

`lv_draw_ppa.h`

`lv_draw_ppa_private.h`

### 4.6.4 EVE External GPU Renderer

EVE is a type of external GPU IC which accepts high-level drawing commands over SPI and outputs the rendered graphics to a display over parallel RGB.

The advantage of using an EVE chip is that the rendering responsibility is removed from the driving MCU so it can be a lower-spec part or dedicate more of its processing time to other tasks. The SPI interface is simpler to connect than parallel RGB especially if the EVE chip is integrated into a display assembly.

LVGL features a renderer for EVE. LVGL UIs can be rendered by EVE and are effectively indistinguishable from the software renderer in most cases.

See also the [FT81x](#) framebuffer driver. It drives the same EVE chips but is a simpler, more standalone implementation which uses software rendering and sends all the pixels over SPI so it is much slower.

## Limitazioni

- Image format, size, and count limit.
- Font format, size, and count limit.
- The total number of tasks rendered per refresh has an upper limit.
- Layers are not supported.

## Utilizzo

### Board Parameters

Find your display parameters and populate a `lv_draw_eve_parameters_t` struct with them.

Here is an example `lv_draw_eve_target_parameters.h` for the Riverdi RVT50HQBNWC00-B which has a BT817Q --- the EVE chip. These parameters were taken from the collection of configs provided by the EVE library supporting this renderer. [See here](#). Check there for your board parameters.

```
lv_draw_eve_parameters_t params = {
    .hor_res = 800,
    .ver_res = 480,
    .vsync0 = 0,
    .vsync1 = 4,
    .voffset = 8,
    .vcycle = 496,
    .hsync0 = 496,
    .hsync1 = 4,
    .hoffset = 8,
    .ncycle = 816,
    .pclk = 3,
    .pclkpol = 1,
    .swizzle = 0,
    .cspread = 0,
    .has_crystal = true,
    .has_gt911 = false,
    .backlight_freq = 4000,
    .backlight_pwm = 128,
};
```

## EVE Chip IO Implementation

The user is required to implement the GPIO and SPI IO functionality. The LVGL EVE renderer will call this callback to perform SPI communication with the EVE chip.

```
static void op_cb(lv_display_t * disp, lv_draw_eve_operation_t operation,
                  void * data, uint32_t length)
{
    /* optional: get the `user_data` parameter you passed to `lv_draw_eve_display_create` */
    void * your_user_data = lv_draw_eve_display_get_user_data(disp);

    switch(operation) {
        case LV_DRAW_EVE_OPERATION_POWERDOWN_SET:
            /* Setting the pin low powers down the EVE chip */
            your_gpio_write(PD_N_PIN, 0);
            break;
        case LV_DRAW_EVE_OPERATION_POWERDOWN_CLEAR:
            /* setting the pin high powers on the EVE chip */
            your_gpio_write(PD_N_PIN, 1);
            break;
        case LV_DRAW_EVE_OPERATION_CS_ASSERT:
            /* setting the pin low asserts the EVE chip SPI device */
            your_gpio_write(CS_N_PIN, 0);
            break;
        case LV_DRAW_EVE_OPERATION_CS_DEASSERT:
            /* setting the pin high de-asserts the EVE chip SPI device */
            your_gpio_write(CS_N_PIN, 1);
            break;
        case LV_DRAW_EVE_OPERATION_SPI_SEND:
            /* `data` is the data to send */
            your_spi_transmit(data, length);
            break;
        case LV_DRAW_EVE_OPERATION_SPI_RECEIVE:
            /* `data` is the destination for the data */
            your_spi_receive(data, length);
            break;
    }
}
```

You will also need to initialize your SPI peripheral and GPIO pins.

22 MHz was the highest SPI speed that worked during testing with the Riverdi board and the ESP32-S3. You may not have success with this speed so it is recommended to validate with an `SPI_SPEED` value of `10` (10 MHz) and increase experimentally in your testing.

## LVGL EVE Display Creation

To create the LVGL display for the EVE renderer, you call `lv_draw_eve_display_create(params, op_cb, your_user_data)` which returns the created display. `your_user_data` can be NULL. It should be called after GPIO and SPI is initialized. You may choose to initialize your IO the first time `op_cb` is called.

No buffers are required for the LVGL EVE renderer because no pixels are written to any buffers in the device running LVGL. When something needs to be drawn, a series of commands are sent to EVE.

## Touch Indev Creation

`lv_draw_eve_touch_create(disp)` creates a touch *Input Device* (`lv_indev`) for the display.

You may need to configure the i2c address of the touch controller connected to EVE. See the section [EVE Register Access](#) for more info about register access.

Here is an example of setting the `REG_TOUCH_CONFIG` register on a BT817q EVE chip for a capacitive touch screen with a controller that has the i2c address `0x15`.

```
/*
15: 0: capacitive, 1: resistive      CAPACITIVE
14: host mode                      NO
13: reserved
12: ignore short circuit protection NO
11: low-power mode                 NO
10-4: 7-bit i2c address            0x15
3: reserved
2: suppress 300ms startup         NO
1-0: 2-bit sampling clocks val    use 1 (the reset default)
*/
lv_draw_eve_memwrite16(disp, LV_EVE_REG_TOUCH_CONFIG, 0x0151);
```

## Rotazione del Display

Efficient display rotation is fully supported through `lv_display_set_rotation()`. Touch input rotation is handled accordingly.

## EVE Register Access

The functions `lv_draw_eve_memread8()`, `lv_draw_eve_memread16()`, `lv_draw_eve_memread32()`, `lv_draw_eve_memwrite8()`, `lv_draw_eve_memwrite16()`, `lv_draw_eve_memwrite32()` are available if needed. They are wrappers around `EVE_memRead8`, etc.

Register definitions and other EVE enumerations are available when you include `lvgl.h` under the prefix namespace `LV_EVE_`. I.e., `REG_ID` is available as `LV_EVE_REG_ID` and `EVE_ROM_CHIPID` is available as `LV_EVE_EVE_ROM_CHIPID`, etc.

## Ulteriori Informazioni

- [https://brtchip.com/wp-content/uploads/Support/Documentation/Programming\\_Guides/ICs/EVE/FT81X\\_Series\\_Programmer\\_Guide.pdf](https://brtchip.com/wp-content/uploads/Support/Documentation/Programming_Guides/ICs/EVE/FT81X_Series_Programmer_Guide.pdf)
- [https://brtchip.com/wp-content/uploads/2024/06/BRT\\_AN\\_033\\_BT81X-Series-Programming-Guide.pdf](https://brtchip.com/wp-content/uploads/2024/06/BRT_AN_033_BT81X-Series-Programming-Guide.pdf)

## 4.6.5 Accelerazione NemaGFX

NemaGFX è un'API grafica di alto livello supportata da molte GPU 2.5D embedded. LVGL può utilizzarla per il rendering nativo della grafica LVGL.

### Iniziare a usare il display Riverdi STM32U5 da 5 pollici

`lv_port_riverdi_stm32u5` è un "porting" pronto all'uso per il display embedded Riverdi STM32 da 5,0" (STM32U599NJH6Q o STM32U5A9NJH6Q) con supporto Nema. Seguire le istruzioni nel file readme per iniziare.

`lv_port_stm32u5g9j-dk2` è un "porting" pronto all'uso per il devkit STM32U5G9J-DK2 con display da 5,0" e GPU NeoChrom VG. Seguire le istruzioni nel file readme per iniziare.

## Utilizzo e Configurazione

Abilitare il renderer impostando `LV_USE_NEMA_GFX` a 1 in `lv_conf.h`. Se si utilizza `LV_USE_NEMA_VG`, impostare `LV_NEMA_GFX_MAX_RESX` e `LV_NEMA_GFX_MAX_RESY` in base alle dimensioni del display che si utilizzerà, in modo che venga riservata sufficiente memoria statica per VG. Senza VG, il renderer software eseguirà più tipi di attività.

"libs/nema\_gfx" contiene i binari precompilati per i driver GPU Nema.

`lv_port_riverdi_stm32u5` è già configurato per linkare i binari "cortex\_m33\_revC" durante la compilazione e `lv_port_stm32u5g9j-dk2` configurato per linkare i binari "cortex\_m33\_NemaPVG" durante la compilazione.

"cortex\_m33\_revC" funziona su tutti i dispositivi STM32 m33 con core Nema, mentre "cortex\_m33\_NemaPVG" supporta inoltre l'accelerazione di tassellazione e moltiplicazione di matrici di cui sono capaci i core Nema U5F e U5G STM32.

Con un diverso progetto STM32CubeIDE, è possibile configurare le librerie da linkare cliccando col pulsante destro del mouse sul progetto nella barra laterale "Project Explorer", selezionando "Properties", andando su "C/C++ Build", "Settings", "MCU G++ Linker" e quindi su "Libraries". Aggiungere una voce in "Libraries (-l)" che sia "nemagfx-float-abi-hard". Aggiungere una voce in "Library search path (-L)" che sia un path per "libs/nema\_gfx/lib/core/cortex\_m33\_revC/gcc" ad esempio "\${workspace\_loc}/\${ProjName}/Middlewares/LVGL/lvgl/libs/nema\_gfx/lib/core/cortex\_m33\_revC/gcc} ". Aggiungere anche la directory "libs/nema\_gfx/include" ai percorsi di ricerca degli include. In "MCU GCC Compiler", "Include paths", aggiungere una voce a "Include paths (-I)" che corrisponde al percorso di "libs/nema\_gfx/include" ad esempio "\${workspace\_loc}/\${ProjName}/Middlewares/LVGL/lvgl/libs/nema\_gfx/include} ". Cliccare su "Apply and Close".

### 1 Nota

```
cannot find -lnemagfx-float-abi-hard: No such file or directory
```

La versione della libreria per ST è stata aggiornata a revC quindi il componente del path `cortex_m33` deve essere aggiornato a `cortex_m33_revC` in STM32CubeIDE.

È supportato `LV_COLOR_DEPTH` a 32 e 16 bit.

Al momento della stesura di questo articolo, il supporto per `LV_USE_OS` è sperimentale e non ancora funzionante in `lv_port_riverdi_stm32u5`

NemaGFX richiede una semplice implementazione HAL per allocare memoria ed eventualmente bloccare le risorse. È possibile utilizzare un'implementazione HAL personalizzata per la propria piattaforma oppure utilizzare una delle implementazioni fornite impostando `LV_USE_NEMA_HAL_CUSTOM`.

## Immagini TSC

Questo motore di rendering può disegnare immagini TSC (ThinkSilicon Compression). I formati colore TSC 4/6/6A/12/12A fanno parte di `lv_color_format_t`. Tutti gli altri motori di rendering ignoreranno le immagini con questi formati colore. Definire una variabile descrittore di immagine con il formato colore TSC corrispondente e la GPU sarà in grado di disegnarla direttamente. Non è necessario specificare Stride perché verrà calcolato dal motore di rendering.

```
const lv_image_dsc_t img_demo_widgets_avatar_tsc6a = {
    .header.cf = LV_COLOR_FORMAT_NEMA_TSC6A,
    .header.w = 144,
    .header.h = 144,
    .data = img_demo_widgets_avatar_tsc6a_map,
    .data_size = sizeof(img_demo_widgets_avatar_tsc6a_map),
};
```

## DMA2D

Il renderer Nema utilizza DMA2D per il flush in parallelo con il rendering in `lv_port_riverdi_stm32u5`.

Se l'STM non ha la GPU Nema, potrebbe comunque supportare DMA2D. DMA2D è una semplice periferica in grado di disegnare riempimenti e immagini indipendentemente dalla CPU. Vedere il [Supporto DMA2D](#) di LVGL.

**API***lv\_draw\_nema\_gfx.h***4.6.6 NXP PXP GPU****API***lv\_draw\_pxp.h***4.6.7 NXP VGLite GPU****API***lv\_draw\_vglite.h***4.6.8 NXP G2D GPU****API***lv\_draw\_g2d.h***4.6.9 Renderer SDL****API***lv\_draw\_sdl.h***4.6.10 STM32 DMA2D GPU****API***lv\_draw\_dma2d.h**lv\_draw\_dma2d\_private.h***4.6.11 GPU Generica VG-Lite**

Si tratta di un'implementazione generica del backend di rendering VG-Lite, progettata per utilizzare l'API generica di VeriSilicon per gestire l'hardware GPU il più possibile.

Anche con diversi produttori di chip, purché utilizzino la stessa versione dell'API VG-Lite come backend di rendering, l'accelerazione del rendering LVGL può essere supportata senza la necessità di adattamenti LVGL.

**Configurazione**

1. Impostare `LV_USE_DRAW_VG_LITE` a 1 in `lv_conf.h` per abilitare il backend di rendering VG-Lite. Assicurarsi che l'hardware sia stato adattato all'API VG-Lite e che sia stato esposto il path assoluto a `vg_lite.h`, a cui lvgl può fare riferimento direttamente.
2. Per confermare il metodo di inizializzazione della GPU, ci sono due modi:
  - L'SDK richiama autonomamente la funzione di inizializzazione della GPU durante l'avvio del sistema e la GPU è disponibile all'avvio di LVGL; impostare `LV_VG_LITE_USE_GPU_INIT` a 0.
  - LVGL richiama attivamente la funzione di inizializzazione della GPU e l'SDK deve implementare la funzione pubblica `gpu_init()`. LVGL la richiamerà per completare l'inizializzazione hardware della GPU durante l'avvio; impostare `LV_VG_LITE_USE_GPU_INIT` a 1.
3. Impostare la configurazione `LV_VG_LITE_USE_ASSERT` per abilitare il controllo dei parametri di chiamata della GPU. A causa della complessità dei parametri utilizzati nelle chiamate della GPU, parametri errati possono causare un funzionamento anomalo dell'hardware della GPU, ad esempio dimenticando di aggiungere un simbolo di fine al path o non rispettando i requisiti di allineamento per il buffer stride. Per risolvere rapidamente tali problemi, è stato aggiunto un rigoroso controllo dei parametri prima di ogni chiamata a VG-Lite,

che include la convalida del buffer stride e il controllo dell'invertibilità della matrice. Quando viene rilevato un parametro di errore, verrà eseguita un'asserzione per visualizzarlo, consentendo all'utente di apportare tempestivamente correzioni e ridurre il tempo sprecato nella simulazione hardware. Si noti che l'abilitazione di questo controllo ridurrà le prestazioni del runtime. Si consiglia di abilitarlo in modalità Debug e di disabilitarlo nella versione Release.

4. Impostare la configurazione `LV_VG_LITE_FLUSH_MAX_COUNT` per specificare il metodo di "flush". VG-Lite utilizza due set di buffer di comando per il rendering delle istruzioni e un utilizzo efficace di questo meccanismo può migliorare notevolmente l'efficienza di disegno. Attualmente sono supportati due metodi di buffering:

- Impostare `LV_VG_LITE_FLUSH_MAX_COUNT` a zero (consigliato). Il backend di rendering otterrà lo stato di funzionamento della GPU ogni volta che scrive istruzioni di rendering nel buffer di comando.

Quando la GPU è inattiva, chiamerà immediatamente `vg_lite_flush` per notificare alla GPU di avviare il rendering e scambiare il buffer di comando. Quando la GPU è occupata, continuerà a riempire la cache del buffer di comando con le istruzioni di rendering. Il driver sottostante determinerà automaticamente se il buffer di comando è stato riempito. Quando sta per essere riempito, attenderà forzatamente la fine delle attività di disegno non completate e scambierà il buffer dei comandi. Questo metodo può migliorare efficacemente l'utilizzo della GPU, soprattutto in scenari in cui si esegue il rendering del testo, poiché il tempo di disegno della GPU e il tempo di preparazione dei dati della CPU sono molto simili, consentendo a CPU e GPU di funzionare in parallelo.

- Set `LV_VG_LITE_FLUSH_MAX_COUNT` to a value greater than zero, such as 8. After writing 8 rendering instructions to the command buffer, the rendering backend will call `vg_lite_flush` to notify the GPU to start rendering and swap the command buffer.

5. Impostare la configurazione `LV_VG_LITE_USE_BOX_SHADOW` per utilizzare il rendering GPU per le ombreggiature. In effetti, l'hardware GPU non supporta effettivamente il rendering delle ombreggiature. Tuttavia, attraverso la sperimentazione, si è scoperto che un effetto ombra simile può essere ottenuto utilizzando più layer di bordi con diversi livelli di trasparenza. Si consiglia di abilitare questa configurazione in scenari in cui i requisiti di qualità delle ombre non sono elevati, in quanto può migliorare significativamente l'efficienza di rendering.
6. Impostare la configurazione `LV_VG_LITE_GRAD_CACHE_CNT` per specificare il numero di voci della cache del gradiente. Il disegno di gradienti include gradienti lineari e radiali. L'utilizzo di una cache può ridurre efficacemente il numero di volte in cui l'immagine del gradiente viene creata e migliorare l'efficienza del disegno. Ogni singolo gradiente consuma circa 4 KB di memoria della GPU. Se nell'interfaccia vengono utilizzati molti gradienti, si può provare ad aumentare il numero di voci nella cache dei gradienti. Se l'API VG-Lite restituisce l'errore `VG_LITE_OUT_OF_RESOURCES`, si può provare ad aumentare la dimensione del pool di memoria della GPU o a ridurre il numero di voci nella cache dei gradienti.
7. Impostare la configurazione `LV_VG_LITE_STROKE_CACHE_CNT` per specificare il numero di cache del path del tratto. Se i parametri del tratto non cambiano, i parametri del tratto generati in precedenza vengono recuperati automaticamente dalla cache per migliorare le prestazioni di rendering. La memoria occupata dal tratto è strettamente correlata alla lunghezza del percorso. Se l'API VG-Lite restituisce l'errore `VG_LITE_OUT_OF_RESOURCES`, si può provare ad aumentare la dimensione del pool di memoria della GPU o a ridurre il numero di voci nella cache dei gradienti.

NOTA: Il backend di rendering VG-Lite non supporta chiamate multi-thread, assicurarsi che `LV_USE_OS` sia sempre configurato come `LV_OS_NONE`.

## VG-Lite Simulator

LVGL integra un simulatore VG-Lite basato su ThorVG. Il suo scopo è semplificare il debug dell'adattamento VG-Lite e ridurre i tempi di debug e individuazione dei problemi sui dispositivi hardware. Per istruzioni dettagliate, vedere [VG-Lite Simulator](#).

## API

`lv_draw_vglite.h`  
`lv_vglite_buf.h`

## 4.7 Framework

### 4.7.1 Arduino

La libreria [LVGL library](#) è disponibile direttamente come libreria Arduino.

Si noti che è necessario scegliere una scheda sufficientemente potente per eseguire LVGL e la propria GUI. Vedere i [requisiti di LVGL](#).

Ad esempio, ESP32 è un buon candidato per creare interfacce utente con LVGL.

Se si utilizza Arduino GIGA Display Shield (la documentazione [qui](#)), utilizzare la libreria [Arduino\\_H7\\_Video](#), inclusa nel pacchetto della scheda Arduino Mbed OS ([qui](#)), che fornisce compatibilità LVGL integrata.

### Ottenere la libreria LVGL per Arduino

LVGL può essere installato tramite l'Arduino IDE Library Manager o come libreria .ZIP.

Si può fare il Download dell'ultima versione di LVGL da GitHub e copiarla semplicemente nella cartella della libreria di Arduino.

### Configurare i driver

Per iniziare, si consiglia di utilizzare la libreria [TFT\\_eSPI](#) come driver TFT per semplificare i test. Per farla funzionare, configurare [TFT\\_eSPI](#) in base al tipo di display TFT modificando:

- `User_Setup.h`
- oppure selezionando una configurazione in `User_Setup_Select.h`

Entrambi i file si trovano nella cartella della libreria [TFT\\_eSPI](#).

### Configurare LVGL

LVGL ha un proprio file di configurazione chiamato `lv_conf.h`. Una volta installato LVGL, seguire questi passaggi di configurazione:

1. Andare alla directory delle librerie installate di Arduino
2. Andare a `lvgl` e copiare `lv_conf_template.h` come `lv_conf.h` nella directory delle Librerie Arduino accanto alla cartella delle librerie di `lvgl`.
3. Aprire `lv_conf.h` e modificare il primo `#if 0` in `#if 1` per abilitare il contenuto del file
4. Impostare la profondità di colore del display in `LV_COLOR_DEPTH`

Infine, il layout con `lv_conf.h` dovrebbe apparire così:

```
arduino
|-libraries
  |-lvgl
  |-other_lib_1
  |-other_lib_2
  |-lv_conf.h
```

### Inizializzare ed eseguire LVGL

Consultare [LVGL\\_Arduino.ino](#) per vedere come inizializzare LVGL. [TFT\\_eSPI](#) viene utilizzato come driver del display.

Nel file INO si può vedere come registrare un display e un touchpad per LVGL e chiamare un esempio.

## Usare gli esempi e le demo

Notare che non esiste un file INO dedicato per ogni esempio. Invece, si può caricare un esempio chiamando la funzione `lv_example_...`. Ad esempio `lv_example_btn_1()`.

### importante

A causa di alcune limitazioni del sistema di compilazione di Arduino, è necessario copiare `lvgl/examples` in `lvgl/src/examples`. Analogamente, per le demo `lvgl/demos` in `lvgl/src/demos`.

## Debug e logging

LVGL può visualizzare informazioni di debug in caso di problemi. Nell'esempio `LVGL_Arduino.ino` è presente il metodo `my_print`, che invia queste informazioni di debug all'interfaccia seriale. Per abilitare questa funzionalità, è necessario modificare il file `lv_conf.h` e abilitare il logging nella sezione `log settings`:

```
/* Log settings */
#define USE_LV_LOG      1 /* Enable/disable the log module */
#if USE_LV_LOG
/* How important log should be added:
 * LV_LOG_LEVEL_TRACE   A lot of logs to give detailed information
 * LV_LOG_LEVEL_INFO    Log important events
 * LV_LOG_LEVEL_WARN    Log if something unwanted happened but didn't cause a problem
 * LV_LOG_LEVEL_ERROR   Only critical issue, when the system may fail
 * LV_LOG_LEVEL_NONE    Do not log anything
*/
#define LV_LOG_LEVEL     LV_LOG_LEVEL_WARN
```

Dopo aver abilitato il modulo di log e impostato `LV_LOG_LEVEL` di conseguenza, il log di output viene inviato alla porta **Seriale** @ 115200 bps.

## 4.7.2 PlatformIO

TODO [da fare]

## 4.7.3 Tasmota e berry

### Cos'è Tasmota?

Tasmota is a widely used open-source firmware for ESP8266 and ESP32 based devices. It supports a wide variety of devices, sensors and integrations to Home Automation and Cloud services. Tasmota firmware is downloaded more than 200,000 times each month, and has an active and growing community.

Tasmota offre accesso a centinaia di dispositivi supportati, supporto completo di MQTT, HTTP(S), integrazione con i principali sistemi di domotica, una miriade di sensori, IR, RF, Zigbee, Bluetooth, AWS IoT, Azure IoT, Alexa e molto altro.

### Cos'è Berry?

Berry è un linguaggio di scripting embedded a tipizzazione dinamica ultraleggero. È progettato per dispositivi embedded a basse prestazioni. L'interprete di Berry include un compilatore one-pass e una VM basata su registri; tutto il codice è scritto in ANSI C99. Berry offre una sintassi molto simile a Python e si ispira alla VM LUA. È completamente integrato in Tasmota.

### Punti di forza di Berry

Berry offre i seguenti vantaggi:

- Leggero: Un interprete ben ottimizzato con risorse molto ridotte. Ideale per l'uso nei microprocessori.
- Veloce: compilatore bytecode ottimizzato a passaggio singolo e macchina virtuale basata su registri.
- Potente: supporta la programmazione imperativa, la programmazione orientata agli oggetti e la programmazione funzionale.
- Flessibile: Berry è uno script di tipo dinamico ed è progettato per l'integrazione nelle applicazioni. Può fornire una buona scalabilità dinamica per il sistema host.

- Semplice: sintassi semplice e naturale, supporto per la garbage collection e interfaccia FFI (foreign function interface) facile da usare.
- Risparmio di RAM: Con la costruzione di oggetti in fase di compilazione, la maggior parte degli oggetti costanti viene memorizzata in segmenti di dati di codice di sola lettura, quindi l'utilizzo di RAM dell'interprete è molto basso all'avvio.

Tutte le funzionalità sono descritte nel [Berry Reference Manual](#)

---

### Perché LVGL + Tasmota + Berry?

Nel 2021, Tasmota ha aggiunto il supporto completo di LVGL per i dispositivi basati su ESP32. Ha inoltre introdotto il linguaggio di scripting Berry, un linguaggio compatto simile a Python e completamente integrato in Tasmota.

È ora disponibile una mappatura completa di LVGL nel linguaggio Berry, simile a quella di MicroPython. Consente di utilizzare oltre il 98% di tutte le funzionalità di LVGL. È anche possibile scrivere widget personalizzati in Berry.

Versioni supportate: LVGL v8.0.2, LodePNG v20201017, Freetype 2.10.4

### Tasmota + Berry + LVGL possono essere utilizzati per:

- Interfaccia grafica per prototipazione rapida.
- Abbreviare il ciclo di modifica e messa a punto dell'interfaccia utente grafica.
- Modellare la GUI in modo più astratto definendo widget compositi riutilizzabili, sfruttando le funzionalità del linguaggio Berry come Ereditarietà, Chiusure, Gestione delle Eccezioni...
- Rendere LVGL accessibile a un pubblico più vasto. Non è necessario conoscere il C per creare una buona GUI su un sistema embedded.

È in fase di sviluppo anche un'interfaccia di livello superiore compatibile con [OpenHASP](#).

---

### Che aspetto ha?

In breve: Simile a MicroPython, è molto simile all'API C, ma orientata agli oggetti per i componenti LVGL.

Immergiamoci subito in un esempio!

### Un semplice esempio

```
lv.start()          # start LVGL
scr = lv.screen_active()    # get default screen
btn = lv.btn(scr)        # create button
btn.center()
label = lv.label(btn)    # create a label in the button
label.set_text("Button") # set a label to the button
```

### Come si usa?

Si può iniziare in meno di 10 minuti su un M5Stack o un dispositivo equivalente in meno di 10 minuti con questo breve tutorial

### Dove si possono trovare maggiori informazioni?

- Documentazione di Tasmota
- Documentazione di Berry
- Documentazione di Berry per Tasmota LVGL

## 4.8 IDE

### 4.8.1 MDK

TODO [da fare]

### 4.8.2 Simulatore su PC

Si può provare LVGL **utilizzando solo il PC** (ovvero senza alcuna scheda di sviluppo). LVGL verrà eseguito in un ambiente di simulazione sul PC, dove chiunque può scrivere e sperimentare applicazioni LVGL reali.

L'utilizzo del simulatore su un PC presenta i seguenti vantaggi:

- Indipendenza dall'hardware: si scrive il codice, lo si esegue sul PC e se ne visualizza il risultato su un monitor.
- Multiplataforma: Qualsiasi sistema Windows, Linux o macOS può eseguire il simulatore per PC.
- Portabilità: Il codice scritto è portatile, il che significa che si può semplicemente copiarlo durante la migrazione su hardware embedded.
- Facile Validazione: Il simulatore è anche molto utile per segnalare bug perché fornisce una piattaforma comune per tutti gli utenti.
- Una migliore esperienza per gli sviluppatori: Su PC i debugger sono solitamente più veloci e migliori, si può loggare su file, aggiungere molte `printf`, eseguire la profilazione e così via.

### Selezionare un IDE

Il simulatore è compatibile con diversi IDE (Integrated Development Environment). Scegli l'IDE preferito, leggerne il suo file README su GitHub, scaricare il progetto e caricarlo nell'IDE.

- [Eclipse con driver SDL](#): Consigliato su Linux e Mac, supporta anche CMake
- [VisualStudio](#): Consigliato su Windows
- [VSCode con driver SDL](#): Consigliato su Linux (SDL) e Mac (SDL)
- [CodeBlocks](#): Consigliato su Windows
- [PlatformIO con driver SDL](#): Consigliato su Linux e Mac, ma supporta anche un ambiente STM32
- [Linux Generico](#): Progetto basato su CMake in cui è possibile passare facilmente da fbdev a DRM e SDL.
- [MDK con FastModel](#): Per Windows

Progetto esterno non gestito dall'organizzazione LVGL:

- [QT Creator](#): Multiplataforma

### Driver integrati

LVGL include diversi *driver integrati*.

Anche se un progetto di simulazione include, ad esempio, SDL, è possibile sostituirlo facilmente abilitando un altro driver in `lv_conf.h` e chiamando la sua funzione `create`.

Ad esempio, per utilizzare il dispositivo frame buffer di Linux al posto di SDL, è sufficiente abilitare `LV_USE_LINUX_FBDEV` e chiamare

```
lv_display_t *display = lv_linux_fbdev_create();
lv_linux_fbdev_set_file(display, "/dev/fb0");
```

## 4.9 (RT)OS

### 4.9.1 Buildroot

Buildroot è un set di Makefile e patch che semplifica e automatizza il processo di creazione di un ambiente Linux completo e avviabile per un sistema embedded, utilizzando al contempo la cross-compilazione per consentire la creazione per più piattaforme target su un singolo sistema di sviluppo basato su Linux.

#### Configurazione Rapida

È disponibile un repository Git che include tutto il necessario per testare la configurazione di Buildroot senza seguire la guida. È pensato per scopi di test e può essere utilizzato se si ha già familiarità con i concetti e l'architettura di Buildroot.

Ottenerne il repository

```
git clone --recurse-submodules https://github.com/lvgl/lv_buildroot.git
```

Creare l'immagine

```
./build.sh
```

A questo punto, si ha un'immagine che può essere flashata su una SD e utilizzata.

Ora compilare l'SDK e installarlo

```
cd output
make sdk
mkdir -p ~/sdk
tar -xzf images/aarch64-buildroot-linux-gnu_sdk-buildroot.tar.gz -C ~/sdk
```

L'SDK è installato. Compilare l'applicazione.

```
cd ../application/lv_benchmark
source ../setup-build-env.sh
cmake -B build -S .
make -j $(nproc) -C build
cd .....
```

Installare il binario in rootfs\_overlay

```
cp application/lv_benchmark/bin/lvgl-app resources/board/rootfs_overlay/usr/bin/
```

Rigenerare l'immagine con l'overlay rootfs aggiornato.

```
cd output
make
```

Andare su [Eseguire il flashing dell'immagine](#) per testare il sistema.

#### Immagine personalizzata RPi4

Questo capitolo offre una guida dettagliata per la creazione di un'immagine personalizzata per Raspberry Pi 4 (RPi4). Verranno introdotti i componenti e i concetti chiave di Buildroot. Il processo descritto qui può essere adattato per l'utilizzo con altre schede. Per funzionalità avanzate non trattate in questa guida, consultare il [Manuale di Buildroot](#)

Questa guida è stata realizzata su Ubuntu 22.04

#### Preparare l'area di lavoro

L'area di lavoro è strutturata per utilizzare Buildroot senza alcuna modifica. Eseguire i seguenti comandi:

```
mkdir rpi4_buildroot_lvgl && cd rpi4_buildroot_lvgl
mkdir resources && cd resources
mkdir board
mkdir configs
touch Config.in && touch external.desc && touch external.mk
echo "name: rpi4_lvgl" >> external.desc
echo "desc: rpi4_lvgl custom build" >> external.desc
cd ..
```

Ogni utilità di gestione cartelle verrà spiegata nel corso della guida.

## Ottenere Buildroot

Innanzitutto, secondo il suo [manuale](#), Buildroot richiede l'installazione di determinati pacchetti prima di avviare la compilazione. Installiamoli utilizzando il gestore pacchetti di Ubuntu.

```
sudo apt install sed make binutils gcc g++ bash patch gzip bzip2 perl tar \
  zip python3 unzip rsync wget libncurses-dev
```

Ora che abbiamo soddisfatto tutti i prerequisiti, scarichiamo una versione stabile di Buildroot.

```
git clone --branch 2024.08 --depth 1 https://github.com/buildroot/buildroot.git
```

## Creare un'immagine RPi4

Vediamo se esiste già una configurazione per RPi4 in Buildroot:

```
cd buildroot
make list-defconfigs | grep rasp
```

È disponibile una build per RPi4 a 62 bit: **raspberrypi4\_64\_defconfig**.

Si possono trovare tutte le configurazioni anche nel repository Buildroot **buildroot > configs**

Iniziare con questa configurazione come base, ma deve essere modificata per soddisfare i requisiti di lvgl. Quindi copiamo questa base e teniamola fuori da Buildroot.

```
# move to the project root
cp buildroot/configs/raspberrypi4_64_defconfig resources/configs/raspberrypi4_lvgl_defconfig
```

Creare un nuovo file **build.sh** nella radice del progetto con questo contenuto:

```
#!/bin/bash
SCRIPTPATH=$( cd "$( dirname "$0" )" >/dev/null 2>&1 ; pwd -P )
cd $SCRIPTPATH
mkdir -p output
cd buildroot
make defconfig O=../output BR2_DEFCONFIG=../resources/configs/raspberrypi4_lvgl_defconfig
```

Rendere questo file eseguibile.

```
chmod +x build.sh
./build.sh
```

L'esecuzione di questo script crea una cartella di output che contiene tutto ciò che il processo genererà:

- Risorse recuperate
- Toolchain
- Immagini

Questo aiuta a mantenere pulita la directory Buildroot. Quando lo script viene eseguito, viene creata una cartella (quasi) vuota denominata **output**, poiché il processo di build non è ancora iniziato.

Modifichiamo la configurazione con **menuconfig**.

```
cd output
make menuconfig
```

Modificare queste configurazioni:

- Nome host di sistema in **lvgl-buildroot** in **system\_configuration > System hostname**.
- Password di root in **lvgl-buildroot** in **system\_configuration > Root password**.
- Abilitare SSH in **target-packages > Networking applications > openssh**

Creare l'immagine

```
make
```

## Cosa è successo?

Esploriamo il contenuto della cartella di output e cosa contiene.

### build

Questa cartella contiene i file intermedi e gli artefatti della build per i vari pacchetti in fase di compilazione. Ogni pacchetto ha una propria sottodirectory all'interno della cartella **build**, dove avviene il processo di compilazione. Include file come file di configurazione, codice sorgente e file oggetto generati durante il processo di compilazione.

### host

La cartella **host** contiene file e binari compilati per il sistema host anziché per il sistema target. Include strumenti e utilità necessari per compilare i pacchetti o per eseguire il sistema di build stesso. Può contenere compilatori, strumenti di compilazione e librerie necessari per supportare il processo di compilazione per il sistema target.

### images

Questa directory contiene le immagini di output finali generate per il sistema target, come immagini del file system, immagini del kernel o immagini del bootloader. A seconda della configurazione, si potrebbero trovare file come **rootfs.tar**, **zImage**, **uImage**, o altri, pronti per essere distribuiti sull'hardware target.

### target

La cartella **target** contiene i file specificamente destinati al sistema di destinazione. Questo include il file system di root e qualsiasi file aggiuntivo che verrà incluso nell'ambiente di destinazione. La struttura all'interno di questa cartella spesso imita la struttura delle directory di un sistema Linux standard, con directory come **bin**, **lib**, **etc**, **usr** e altre, che contengono i file binari, le librerie, i file di configurazione e altri componenti necessari per il corretto funzionamento del sistema target.

## Eseguire il flashing dell'immagine

Inserire la scheda SD nel laptop e controllarne il punto di montaggio. In genere è etichettata come **sda** o **sdb**, ma si può usare il comando **lsblk** per confermare questa informazione.

Se è montata su **/dev/sda**, si esegue il seguente comando:

```
sudo dd if=images/sdcard.img of=/dev/sda
```

Come indicato in **images**, l'immagine di output si trova in **images** e si chiama **sdcard.img**.

Collegare un cavo Ethernet al RPi4 e assicurarsi che il laptop e l'RPi4 siano sulla stessa rete.

Si può usare **ifconfig** o **ip a** per trovare il proprio indirizzo IP. Poi, si usa **nmap** o qualsiasi altro strumento per scansionare la rete e trovare l'indirizzo IP del RPi4.

```
nmap -sn <YOUR_IP_ADDRESS>/24 | grep lvgl
```

Tutto questo può essere fatto con l'UART invece che con SSH se non si vuol connettere l'RPi4 alla rete.

## Applicazione LVGL

Questa sezione fornisce informazioni sui passaggi da seguire per ottenere un'applicazione personalizzata che utilizzi LVGL in esecuzione sulla scheda.

## Aggiornare RootFS

A seconda dell'applicazione, potrebbe essere necessario aggiornare il rootfs. Prendiamo come esempio la compilazione di LVGL con DRM. Il sistema deve avere **libdrm** installato.

```
cd output  
make menuconfig
```

Per cercare un pattern di stringhe nella configurazione, premere / seguito dal pattern desiderato. Ad esempio, cercare `libdrm`. Si dovrebbe trovare `BR2_PACKAGE_LIBDRM` impostato su `[=n]`. Il campo "Location" indica dove trovare questa opzione. Premendo il tasto numerico corrispondente (9-0), è possibile accedere direttamente all'opzione.

Nella finestra **Search Results**, la sezione **Depends on** elenca i pacchetti o le opzioni richiesti che devono essere abilitati (o disabilitati) per rendere configurabile il pacchetto target. Per qualsiasi pacchetto che si desidera aggiungere, queste dipendenze devono essere soddisfatte; in caso contrario, l'opzione rimarrà nascosta.

Una volta completata la configurazione del pacchetto, si compila l'ambiente per aggiungere i pacchetti (`make` nella cartella **output**).

Per verificare che la libreria sia stata installata, possiamo trovarla nella sysroot di destinazione:

```
find build/ -name "*libdrm*"
```

Si dovrebbero vedere la cartella "include" e i file .so.

## Generare l'SDK e configurare l'ambiente

Generare un SDK utilizzabile per la cross-compilazione dell'applicazione per il target (RPi4).

make sdk

Viene generato un file `.tar.gz` in `output/images`. Questo è l'SDK!

Per utilizzarlo, estrarlo ovunque.

```
mkdir -p ~/sdk  
tar -xzf images/aarch64-buildroot-linux-gnu_sdk-buildroot.tar.gz -C ~/sdk  
cd ..
```

Per configurare l'ambiente

```
mkdir application && cd application  
touch setup-build-env.sh && chmod +x setup-build-env.sh
```

Lo script `setup-build-env.sh` si presenta così:

```
#!/bin/bash

SDK_PATH="$HOME/sdk/aarch64-buildroot-linux-gnu_sdk-buildroot"
export PATH="${SDK_PATH}/bin:${PATH}"

export SYSROOT="${SDK_PATH}/aarch64-buildroot-linux-gnu/sysroot"

export CROSS_COMPILE="aarch64-buildroot-linux-gnu-"

export CC="${CROSS_COMPILE}gcc"
export CXX="${CROSS_COMPILE}g++"
export LD="${CROSS_COMPILE}ld"
export AR="${CROSS_COMPILE}ar"
export AS="${CROSS_COMPILE}as"

export CFLAGS="--sysroot=${SYSROOT}"
export LDFLAGS="--sysroot=${SYSROOT}"
```

## Build dell'applicazione

L'ambiente è ora configurato e siamo pronti per compilare un'applicazione utilizzando il repository `lv_benchmark` ispirato a `lv_port_linux`.

Tornare alla radice del progetto e clonare il repository:

```
git clone --recurse-submodules https://github.com/EDGEMTech/lv_benchmark.git
```

L'applicazione è configurata per essere eseguita su fbdev. È possibile mantenere la configurazione di default o modificarla in base alle proprie preferenze.

## Compilare l'applicazione

```
cd lv_benchmark
cmake -B build -S . -DCMAKE_C_COMPILER=${CROSS_COMPILE}gcc -DCMAKE_CXX_COMPILER=${CROSS_COMPILE}g++ -DCMAKE_SYSROOT=${SYSROOT} -DCMAKE_C_FLAGS="--sysroot=${SYSROOT}" -DCMAKE_CXX_FLAGS="--sysroot=${SYSROOT}"
make -j $(nproc) -C build
```

Verificare che l'eseguibile di output sia stato compilato con la toolchain corretta:

```
file bin/lvgl-app
cd .....
```

L'output dovrebbe contenere queste informazioni:

- ARM aarch64
- interpreter /lib/ld-linux-aarch64.so.1

### Impostare un overlay rootfs

In Buildroot, un overlay rootfs (o overlay del filesystem di root) è un meccanismo che consente di aggiungere file, directory e configurazioni personalizzate direttamente nel filesystem di root dell'immagine target durante il processo di build. È un modo per estendere o modificare il contenuto del filesystem di root senza alterare i pacchetti di base o ricompilare tutto.

Create the rootfs overlay structure and include the executable of the built application.

```
mkdir -p resources/board/rootfs_overlay/usr/bin
cp application/lv_benchmark/bin/lvgl-app resources/board/rootfs_overlay/usr/bin/
```

Aggiornare la configurazione.

```
cd output
make menuconfig
```

Trovare l'opzione dell'overlay rootfs **System configuration > Root filesystem overlay directories** e utilizzare il path relativo all'overlay rootfs con l'applicazione (**./resources/board/rootfs\_overlay**) e salvare la configurazione.

Ora utilizzare il comando **make** per applicare l'overlay rootfs e verificare che la sysroot sia stata aggiornata come previsto.

```
find . -name lvgl-app
```

*Eseguire il flashing dell'immagine* ed eseguire l'applicazione di benchmark sulla board.

### 4.9.2 FreeRTOS

TODO [da fare]

### 4.9.3 MQX RTOS

Consultare la homepage di MQX RTOS

TODO [da fare]

### 4.9.4 NuttX RTOS

#### Cos'è NuttX?

NuttX è un sistema operativo real-time (RTOS) maturo e sicuro, con particolare attenzione alla conformità agli standard tecnici e alle dimensioni ridotte. È scalabile da microcontrollori e microprocessori da 8 bit a 64 bit ed è conforme agli standard Portable Operating System Interface (POSIX) e American National Standards Institute (ANSI), nonché a molti sottosistemi simili a Linux. Il modo migliore per pensare a NuttX è considerarlo come un piccolo Unix/Linux per microcontrollori.

## Punti di Forza di NuttX

- **Piccolo** - Si adatta e funziona su microcontrollori con soli 32 kB di Flash e 8 kB di RAM.
- **Conforme** - Si impegna a essere il più compatibile possibile con POSIX e Linux.
- **Versatile** - Supporta numerose architetture (ARM, ARM Thumb, AVR, MIPS, OpenRISC, RISC-V a 32 e 64 bit, RX65N, x86-64, Xtensa, Z80/Z180, ecc.).
- **Modulare** - Il suo design modulare consente agli sviluppatori di selezionare solo ciò che conta davvero e di utilizzare i moduli per includere nuove funzionalità.
- **Popolare** - NuttX è utilizzato da molte aziende in tutto il mondo. Probabilmente è già stato utilizzato un prodotto con NuttX senza sapere che eseguiva NuttX.
- **Prevedibile** - NuttX è un kernel "Realtime preemptible", quindi lo si può usare per creare applicazioni prevedibili per il controllo in tempo reale.

## Perché NuttX + LVGL?

Sebbene NuttX abbia una propria libreria grafica chiamata [NX](#), LVGL è una buona alternativa perché gli utenti potrebbero trovare demo più accattivanti e riutilizzare il codice di progetti precedenti. LVGL è una libreria GUI di alto livello [Basata su Componenti Object-Oriented](#), che potrebbe adattarsi perfettamente a un RTOS con funzionalità avanzate come NuttX. LVGL è implementato in C e le sue API sono in C.

## Ecco alcuni vantaggi dell'utilizzo di LVGL in NuttX

- Sviluppare prima la GUI in Linux e, una volta completata, compilarla per NuttX. Niente di più, nessuna perdita di tempo.
- Di solito, lo sviluppo di GUI per RTOS di basso livello richiede più iterazioni per ottenere il risultato desiderato, dove ogni iterazione consiste in **Modifica del codice > Build > Flash > Run**. Utilizzando LVGL, Linux e NuttX si può ridurre questo processo e testare tutto sul computer e, una volta completato, compilarlo su NuttX e il gioco è fatto.

## NuttX + LVGL potrebbero essere utilizzati per

- Delle demo della GUI per dimostrare le capacità grafiche della scheda.
- Prototipazione rapida della GUI per la presentazione MVP (Minimum Viable Product).
- Visualizzazione dei dati dei sensori direttamente e facilmente sulla scheda senza utilizzare un computer.
- Prodotti finali con una GUI senza touchscreen (ad esempio, un'interfaccia per stampante 3D che utilizza un encoder rotativo per l'inserimento dei dati).
- Prodotti finali con touchscreen (e tutti i tipi di ammennicoli).

## Come iniziare con NuttX e LVGL?

Ci sono molte board nella [NuttX mainline](#) con supporto per LVGL. Usiamo la [STM32F429IDISCOVERY](#) come esempio, perché è una scheda molto popolare.

## Per prima cosa si devono installare i prerequisiti sul sistema.

Utilizziamo Windows Subsystem for Linux

```
$ sudo apt-get install automake bison build-essential flex gcc-arm-none-eabi gperf git libncurses5-dev libtool libusb-dev libusb-1.0.0-dev pkg-config kconfig-frontends openocd
```

Ora creiamo un'area di lavoro in cui salvare i nostri file.

```
$ mkdir ~/nuttxspace
$ cd ~/nuttxspace
```

## Cloniamo i repository NuttX e Apps.es

```
$ git clone https://github.com/apache/incubator-nuttx nuttx
$ git clone https://github.com/apache/incubator-nuttx-apps apps
```

## Simulatore NuttX

Il simulatore NuttX consente di eseguire NuttX e LVGL sul PC. Questo può essere particolarmente utile per sviluppare interfacce utente LVGL con feedback visivo immediato e anche per sviluppare app NuttX indipendenti dalla piattaforma.

## Configurare NuttX per utilizzare la board sim e LVGL Demo

```
$ ./tools/configure.sh sim:lvgl_fb
$ make
```

Se tutto è andato bene, si avrà il file **nuttx**

```
$ ls -l nuttx
-rwxr-xr-x 1 root root 2819920 May 12 15:01 nuttx
```

## Esecuzione del Simulatore NuttX

```
$ ./nuttx
```

Using the **NSH>** terminal start the LVGL demo:

```
nsh> lvldemo
```

## NuttX On-Device

Se non si ha una STM32F429I-DISCO, questa procedura sarà simile per altre schede.

## Configurare NuttX per utilizzare la scheda stm32f429i-disco e la demo LVGL

```
$ ./tools/configure.sh stm32f429i-disco:lvgl
$ make
```

Se tutto è andato a buon fine, si dovrebbe avere il file **nuttx.bin** da flashare sulla scheda:

```
$ ls -l nuttx.bin
-rw-rw-r-- 1 alan alan 287144 Jun 27 09:26 nuttx.bin
```

## Flashare il firmware sulla scheda usando OpenOCD

```
$ sudo openocd -f interface/stlink-v2.cfg -f target/stm32f4x.cfg -c init -c "reset halt" -c "flash write_image erase nuttx.bin 0x08000000"
```

Resetare la scheda e, usando il terminale '**NSH>**', avviare la demo LVGL:

```
nsh> lvldemo
```

## Utilizzo Personalizzato

L'app di esempio chiamata **lvldemo** usa l'integrazione NuttX di LVGL per eseguire le demo in NuttX.

<https://github.com/apache/nuttx-apps/tree/master/examples/lvldemo>

Viene utilizzata da tutte le configurazioni di LVGL. Vedere **sim:lvgl\_fb** e **sim:lvgl\_lcd**.

La si può usare come riferimento per aggiungere LVGL alla propria app NuttX. Iniziamo copiando il contenuto di `lvgldemo.c`. Gestisce i driver dei display basati su fb e lcd, il driver di input touchscreen e libuv.

## Supporto Driver NuttX

- **fbdev** - LVGL può creare un display per i dispositivi `/dev/fb*`.
- **lcd** - LVGL può creare un display per i dispositivi `/dev/lcd*`.
- **input/uinput touchscreen** - LVGL può creare uno o più indev per dispositivi touch che utilizzano il sistema input/uinput NuttX.
- **libuv** - LVGL supporta l'utilizzo di libuv come "event loop manager". Le app libuv possono integrarsi perfettamente.

Non è ancora disponibile il supporto per le tastiere input/uinput. Il driver della tastiera sim NuttX X11 emette codici tasto non standard, quindi l'integrazione di LVGL con esso non sarebbe generica, se fosse aggiunto.

## Altre integrazioni

- Integrazione del profiler LVGL+NuttX
- Heap opzionale della cache immagini dedicato

## Configurazioni

La libreria LVGL è una "app" di NuttX. È possibile configurare LVGL in `menuconfig`.

```
$ ./tools/configure.sh stm32f429i-disco:lvgl
$ make menuconfig
```

Il file **Kconfig** di LVGL fornisce le opzioni di configurazione disponibili per il sistema di build di NuttX. Le configurazioni LVGL in `menuconfig` si trovano in **Application Configuration > Graphics Support > Light and Versatile Graphic Library (LVGL) > LVGL configuration**.

Per trovare la posizione di una configurazione specifica, premere il tasto "/" per aprire un'interfaccia di ricerca.

Ecco alcune configurazioni utilizzabili per personalizzare la configurazione di NuttX e LVGL:

- **LV\_USE\_NUTTX\_INDEPENDENT\_IMAGE\_HEAP** - È possibile abilitare o disabilitare l'heap delle immagini LVGL in NuttX. Per default, è disabilitato. Se lo si abilita, LVGL utilizzerà invece l'heap NuttX.

Utilizzare `LV_STDLIB_LIB` per `LV_USE_STDLIB_MALLOC`, `LV_USE_STDLIB_STRING` o `LV_USE_STDLIB_SPRINTF` è ragionevole, poiché NuttX implementa completamente queste API della libreria standard. La scelta di utilizzare `malloc` di NuttX dipende dal fatto che si desideri che LVGL allochi dall'heap globale di NuttX o utilizzi il proprio.

## Dov'è LV\_OS\_NUTTX?

NuttX cerca di essere compatibile con POSIX ove possibile, il che significa che supporta i pthread (thread POSIX). Per abilitare le funzionalità del sistema operativo in LVGL su NuttX, impostare `LV_USE_OS` a `LV_OS_PTH-READ`. Il motivo principale per cui si abilitano le funzionalità del sistema operativo è il rendering multi-core. Vedere `LV_DRAW_SW_DRAW_UNIT_CNT`. Altrimenti non è necessario impostarlo.

## Utilizzo di una Versione Specifica di LVGL

Ecco i passaggi per utilizzare l'ultima versione (o qualsiasi) di LVGL in NuttX.

Assicurarsi innanzitutto che gli artefatti di build siano stati cancellati. Questo rimuoverà il file LVGL e lo ZIP decompressi esistenti, se presenti.

```
$ make distclean
```

Ora si può usare `git clone` per l'ultima versione di LVGL nell'albero delle app di NuttX. Persisterà anche dopo `make distclean` perché il sistema di build non rimuoverà lvgl se si tratta di un repository Git.

```
$ cd /path/to/nuttxspace/apps/graphics/lvgl
$ git clone https://github.com/lvgl/lvgl.git
```

Infine, si deve copiare il contenuto di `apps/graphics/lvgl/lvgl/Kconfig` al centro di `apps/graphics/lvgl/Kconfig`. Vedere la sezione "PASTE THE CONTENTS ..." qui sotto.

#### `apps/graphics/lvgl/Kconfig`:

```
# For a description of the syntax of this configuration file,
# see the file kconfig-language.txt in the NuttX tools repository.
#
menuconfig GRAPHICS_LVGL
    bool "Light and Versatile Graphic Library (LVGL)"
    default n
    ---help---
        Enable support for the LVGL GUI library.

if GRAPHICS_LVGL
    (PASTE THE CONTENTS OF apps/graphics/lvgl/lvgl/Kconfig HERE)

config LV_OPTLEVEL
    string "Customize compilation optimization level"
    default ""

endif # GRAPHICS_LVGL
```

#### Dove si possono trovare maggiori informazioni?

- Questo post del blog: [LVGL on LPCXpresso54628](#)
- Mailing list NuttX: [Apache NuttX Mailing List](#)

### 4.9.5 PX5 RTOS

Vedere la [homepage](#) di PX5 RTOS

TODO [da fare]

### 4.9.6 QNX

#### Cos'è QNX?

QNX è un sistema operativo commerciale rilasciato per la prima volta nel 1980. Il sistema operativo si basa su un design a microkernel, con file system, stack di rete e vari altri driver, ognuno dei quali eseguito in un proprio processo con uno spazio di indirizzamento separato.

Per maggiori dettagli, consultare [www.qnx.com](http://www.qnx.com).

#### Punti di forza di QNX

- Solo a 64 bit, funziona su x86\_64 e ARMv8
- Richiede una MMU poiché la progettazione impone la separazione tra i processi
- Supporto per migliaia di processi e milioni di thread
- Fino a 64 core, fino a 16 TB di RAM
- Supporto per la virtualizzazione (come host e guest)
- Compatibilità POSIX completa
- Certificazione di sicurezza secondo vari standard automobilistici, industriali e medicali

#### Come eseguire LVGL su QNX?

Esistono due modi per utilizzare LVGL nei progetti QNX. Il primo è simile a come LVGL viene utilizzato su altri sistemi. Il secondo consiste nel compilare LVGL come libreria 'shared' o statica.

## Includere LVGL nel Progetto

Seguire le istruzioni generiche per iniziare a usare LVGL. Dopo aver copiato `lv_conf_template.h` in `lv_conf.h` si apportano le seguenti modifiche a quest'ultimo:

1. Abilitare il supporto QNX:

```
#define LV_USE_QNX 1
```

2. Impostare la profondità di colore a 32:

```
#define LV_COLOR_DEPTH 32
```

3. (Facoltativo) Abilitare il doppio buffering:

```
#define LV_QNX_BUF_COUNT 2
```

## Compilare LVGL come libreria

**Notare che questo metodo è un'alternativa all'inclusione di LVGL nel progetto. Se si sceglie di compilare una libreria, non è necessario seguire le istruzioni della sezione precedente.**

La directory di livello superiore `qnx` include un file make ricorsivo per compilare LVGL, sia come libreria condivisa che come libreria statica per le architetture supportate. Per compilare tutte le librerie, è sufficiente richiamare `make` in questa directory:

```
# cd $(LVGL_ROOT)/env_support/qnx
# make
```

Se si preferisce compilare per un'architettura e una variante specifiche, si va alla directory appropriata e si esegue `make` da lì. Ad esempio, per creare una libreria condivisa per ARMv8:

```
# cd $(LVGL_ROOT)/env_support/qnx/aarch64/so.lib
# make
```

Come regola generale, se si desidera avere una sola applicazione LVGL nel sistema, è meglio utilizzare una libreria statica. Se ce n'è più di una, e soprattutto se vengono eseguite contemporaneamente, è meglio utilizzare la libreria condivisa.

Prima di creare la libreria, è consigliabile modificare `$(LVGL_ROOT)/env_support/qnx/lv_conf.h`, ad esempio per aggiungere font o disabilitare il doppio buffering.

## Scrivere un'Applicazione LVGL

Per creare un'applicazione LVGL per QNX, seguire questi passaggi nel codice:

1. Inizializzare la libreria.
2. Creare una finestra.
3. Aggiungere i dispositivi di input.
4. Creare l'interfaccia utente.
5. Eseguire il "loop" di eventi.

I passaggi 2, 3 e 5 utilizzano chiamate specifiche per QNX, ma il resto del codice dovrebbe essere identico a quello di un'applicazione LVGL scritta per qualsiasi altra piattaforma.

Il codice seguente mostra come creare un'applicazione "Hello World":

```
#include <lvgl.h>

int
main(int argc, char **argv)
{
    /* Initialize the library. */
    lv_init();

    /* Create a 800x480 window. */
    lv_display_t *disp = lv_qnx_window_create(800, 480);
    lv_qnx_window_set_title(disp, "LVGL Example");
```

(continues on next page)

(continua dalla pagina precedente)

```

/* Add keyboard and mouse devices. */
lv_qnx_add_keyboard_device(display);
lv_qnx_add_pointer_device(display);

/* Generate the UI. */
lv_obj_set_style_bg_color(lv_screen_active(), lv_color_hex(0x003a57), LV_PART_MAIN);

lv_obj_t * label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "Hello world");
lv_obj_set_style_text_color(lv_screen_active(), lv_color_hex(0xffffffff), LV_PART_MAIN);
lv_obj_align(label, LV_ALIGN_CENTER, 0, 0);

/* Run the event loop until it exits. */
return lv_qnx_event_loop(display);
}

```

## Building dell'Applicazione

Il building dell'applicazione consiste nel compilare il codice sorgente con gli header LVGL e poi nel linkarlo alla libreria. Questo può essere fatto in molti modi, utilizzando diversi sistemi di build. Di seguito è riportato un semplice file make per l'esempio precedente, che compila per ARMv8 con la libreria "shared":

```

CC=qcc -Vgcc_ntoaarch64le
LVGL_ROOT=$(HOME)/src/lvgl
CFLAGS=-I$(LVGL_ROOT)/env_support/qnx -I$(LVGL_ROOT)
LDFLAGS=lscreen -llvgl -L$(LVGL_ROOT)/env_support/qnx/aarch64/so.lib

lvgl_example: lvgl_example.c
$(CC) $(CFLAGS) -Wall -o $@ $< $(LDFLAGS)

clean:
rm -f *.o *~ lvgl_example

```

## 4.9.7 RT-Thread RTOS

### Cos'è RT-Thread?

RT-Thread è un sistema operativo real-time (RTOS) [open source](#), neutrale e basato sulla community. RT-Thread è disponibile in **versione Standard** e **versione Nano**. Per i sistemi a microcontrollore (MCU) con risorse limitate, la versione Nano, che richiede solo 3 KB di memoria Flash e 1,2 KB di RAM, può essere personalizzata con strumenti di facile utilizzo. Per i dispositivi IoT ricchi di risorse, RT-Thread può utilizzare lo strumento di gestione dei **pacchetti software online**, insieme agli strumenti di configurazione del sistema, per ottenere un taglio modulare rapido e intuitivo, importare senza problemi pacchetti software complessi e ottenere così funzioni complesse come l'interfaccia grafica di Android e gli effetti di scorrimento touch, gli effetti di interazione vocale intelligente e così via.

### Caratteristiche Principali

- Progettato per dispositivi con risorse limitate, il kernel minimo richiede solo 1,2 KB di RAM e 3 KB di Flash.
- Una varietà di interfacce standard, come POSIX, CMSIS, ambiente applicativo C++.
- Ha componenti ricchi e un [ecosistema di pacchetti](#) prospero e in rapida crescita
- Stile di codice elegante, facile da usare, leggere e padroneggiare.
- Elevata scalabilità. RT-Thread ha un'architettura software scalabile di alta qualità, "loose coupling" [accoppiamento libero], modularità, è facile da personalizzare ed espandere.
- Supporta applicazioni ad alte prestazioni.
- Supporta tutti i principali strumenti di compilazione come GCC, Keil e IAR.
- Supporta un'ampia gamma di [architetture e chip](#)

### Come eseguire LVGL su RT-Thread?

 [documentazione cinese]

LVGL è registrato come [software package](#) di RT-Thread. Utilizzando [Env tool](#) o [RT-Thread Studio IDE](#), gli utenti di RT-Thread possono scaricare facilmente il codice sorgente di LVGL e integrarlo nel progetto RT-Thread.

La community RT-Thread ha portato LVGL su diversi BSP [Board Support Package]:

BSP	BSP
Simulatore QEMU	Infineon psoc6-evaluationkit-062S2
Simulatore di Visual Studio	Renesas ra6m3-ek
Nuvoton numaker-iot-m487	Renesas ra6m4-cpk
Nuvoton numaker-pfm-m487	Renesas ra6m3-hmi
Nuvoton nk-980iot	STM32H750 ART-Pi
Nuvoton numaker-m2354	STM32F469 Discovery
Nuvoton nk-n9h30	STM32F407 explorer
Nuvoton numaker-m032ki	STM32L475 pandora
Nuvoton numaker-hmi-ma35d1	NXP imxrt1060-evk
Nuvoton numaker-iot-m467	Raspberry PICO
Nuvoton numaker-m467hj	NXP LPC55S69
synwit swm341	

## Tutorial

- Introduzione a RT-Thread e a come eseguire LVGL su RT-Thread nei simulatori
- Come importare un progetto BSP con il codice più recente in RT-Thread Studio
- Come usare LVGL con RT-Thread Studio nella board STM32F469 Discovery
- Canale YouTube di RT-Thread
- Centro di documentazione di RT-Thread

## 4.9.8 Torizon OS

Torizon OS è una piattaforma software open source che mira a semplificare lo sviluppo e la manutenzione di software Linux embedded.

Utilizzando i container Docker, consente agli sviluppatori di concentrarsi sull'applicazione anziché creare e gestire una distribuzione personalizzata.

Questa guida spiega come creare un'immagine Docker contenente LVGL e una semplice applicazione demo che può essere distribuita su qualsiasi dispositivo Toradex con TorizonOS.

### Prerequisiti

Per seguire questa guida è ovviamente necessario disporre di un SoM Toradex e di una board carrier. Ulteriori informazioni sono disponibili sul [sito web](#) di Toradex.

#### Nota

È necessario avere familiarità con alcuni principi fondamentali di Docker per poter comprendere e seguire questa guida. Inoltre, Docker deve essere presente sull'host di sviluppo.

La documentazione di Toradex è una risorsa utile. Questo articolo contiene molti riferimenti ad essa.

### Configurazione della scheda

Si inizia installando TorizonOS utilizzando Toradex Easy Installer.. Seguire le [guide](#) ufficiali di avvio.

Nell'Easy Installer, selezionare TorizonCore come sistema operativo da installare.

Copiare l'indirizzo IP assegnato al dispositivo dal server DHCP sulla propria rete. L'indirizzo viene visualizzato nell'angolo in basso a destra. Verrà utilizzato in seguito.

Una volta completata la configurazione, il dispositivo si avvierà in TorizonOS.

## Creazione dell'immagine Docker

Toradex fornisce un'estensione [VSCode](#) che offre una raccolta di template utilizzati per configurare e automatizzare le attività necessarie per la cross-compilazione delle applicazioni e la creazione di immagini Docker.

Tuttavia, questa guida spiega come eseguire manualmente tali operazioni.

### Nota

Il template per un'applicazione LVGL è attualmente in fase di aggiunta all'estensione VSCode e sarà disponibile a breve.

Per creare un container Torizon, è necessario abilitare l'emulazione Docker ARM digitando i seguenti comandi:

```
docker run --rm -it --privileged torizon/binfmt
docker run --rm -it --pull always --platform linux/arm64/v8 debian arch
```

Il secondo comando dovrebbe stampare `aarch64`.

Ulteriori informazioni sull'ambiente di build per i container Torizon sono disponibili [qui](#).

Iniziamo creando una directory che conterrà il `Dockerfile` e il codice sorgente che verrà utilizzato per creare l'immagine.

Digitare i seguenti comandi in una finestra di terminale:

```
mkdir -p ~/lvgl_torizon_os/
cd ~/lvgl_torizon_os/
touch Dockerfile

git clone --depth 1 https://github.com/lvgl/lv_port_linux
git -C lv_port_linux submodule update --init
```

Questi comandi creano la directory del progetto e il `Dockerfile`. `git` viene utilizzato per scaricare i repository `lv_port_linux` e `lvgl` da Github.

### Nota

Per default, `lv_port_linux` è configurato per utilizzare il dispositivo framebuffer legacy `/dev/fb0`. È anche possibile utilizzare un altro backend di rendering abilitando le opzioni corrette in `lv_port_linux/lv_conf.h`.

Ora si modifica il `Dockerfile`. Copiare e incollare il blocco seguente nel file:

```
ARG CROSS_SDK_BASE_TAG=3.2.1-bookworm
ARG BASE_VERSION=3.2.1-bookworm
##
## Board architecture
## arm or arm64
##
ARG IMAGE_ARCH=arm64

##
## Directory of the application inside container
##
ARG APP_ROOT=/usr/lvgl_application

# BUILD -----
FROM torizon/debian-cross-toolchain-${IMAGE_ARCH}:${CROSS_SDK_BASE_TAG} AS build

ARG APP_ROOT
ARG IMAGE_ARCH

RUN apt-get -q -y update && \
    apt-get -q -y install && \
    apt-get clean && apt-get autoremove && \
    apt-get install -q -y curl git cmake file && \
    rm -rf /var/lib/apt/lists/*

COPY . ${APP_ROOT}
WORKDIR ${APP_ROOT}

# Compile lv_port_linux
RUN CC=aarch64-linux-gnu-gcc cmake -S ./lv_port_linux -B build
RUN make -j 4 -C ${APP_ROOT}/build

# DEPLOY -----
FROM --platform=linux/${IMAGE_ARCH} torizon/debian:${BASE_VERSION} AS deploy

ARG IMAGE_ARCH
ARG APP_ROOT
```

(continues on next page)

(continua dalla pagina precedente)

```
RUN apt-get -y update && apt-get install -y --no-install-recommends \
&& apt-get clean && apt-get autoremove && rm -rf /var/lib/apt/lists/*
# Copy the lvglsim executable compiled in the build step to the $APP_ROOT directory
# path inside the container
COPY --from=build ${APP_ROOT}/lv_port_linux/bin/lvgl${SIM} ${APP_ROOT}
# Command executed during runtime when the container starts
ENTRYPOINT [ "./lvgl${SIM}" ]
```

The **Dockerfile** acts like a recipe to build two images: **build** and **deploy**.

Per prima cosa scaricare i pacchetti necessari per compilare il simulatore utilizzando il gestore di pacchetti Debian **apt-get**.

Dopo la compilazione, l'eseguibile risultante viene scritto in **lv\_port\_linux/bin/lvgl\${SIM}**.

L'immagine **deploy** verrà distribuita sul dispositivo. L'eseguibile creato nell'immagine precedente viene copiato nella directory **/usr/bin** dell'immagine corrente.

Questo crea un'immagine più piccola che non include la tool chain e le dipendenze di building.

Le immagini vengono compilate con il seguente comando:

```
docker build . -t lvgl_app
```

Docker interpreterà il **Dockerfile** presente nella directory di lavoro corrente. L'argomento **-t** assegna un nome all'immagine risultante.

Al termine, assicurarsi che l'immagine sia elencata da Docker:

```
docker image list | grep lvgl_app
```

Dovrebbe visualizzare l'immagine insieme al suo ID che verrà utilizzato in seguito.

## Deploy dell'immagine del container sul dispositivo

L'immagine è ora pronta per essere distribuita sul dispositivo. Esistono diversi modi per eseguire questa operazione.

Leggere [questo articolo](#) per maggiori informazioni.

In questa guida, configureremo un "container registry Docker" sull'host di sviluppo, accessibile da qualsiasi dispositivo della LAN. La scheda Toradex, essendo sulla stessa rete, sarà in grado di estrarre l'immagine dal registry.

Il registry viene creato come segue:

```
docker run -d -p 5000:5000 --name registry registry:2.7
```

Il flag **-d** esegue il container in modalità "detached". L'argomento **-p** specifica il mapping delle porte. Il "registry container" ascolterà sulla porta **TCP/5000** e verrà mappato esternamente alla stessa porta.

Eseguire il push dell'immagine creata nel passaggio precedente nel registry appena creato:

```
docker tag <IMAGE_ID> 127.0.0.1:5000/lvgl-app
```

```
docker push 127.0.0.1:5000/lvgl-app
```

Per default un "container registry" locale utilizza HTTP in chiaro, quindi l'istanza Docker in esecuzione sul dispositivo deve essere configurata per consentire il recupero di immagini da un repository 'non sicuro'.

Ottenerne l'indirizzo IP dell'host di sviluppo e aprire una shell remota sul dispositivo.

```
sudo su # When prompted type in the password of the torizon user
# Be sure to replace set the IP address of your host instead
cat <> heredoc > /etc/docker/daemon.json
{
    "insecure-registries" : ["<IP_ADDR_OF_DEVELOPMENT_HOST>:5000"]
}
heredoc

systemctl restart docker.service

# Return to regular user
exit

# Pull the image from your development host
docker pull <IP_ADDR_OF_DEVELOPMENT_HOST>:5000/lvgl-app
```

Per default, i container Docker sono isolati dall'hardware del sistema host. Sebbene questo sia ottimo per la sicurezza, un'applicazione embedded avrà ovviamente bisogno di [accesso all'hardware](#).

Il container che esegue l'applicazione LVGL necessita di accesso al framebuffer `/dev/fb0` del dispositivo.

Utilizzando l'argomento `--device` è possibile mappare un dispositivo a un container.

Avviare il container in questo modo:

```
docker run --device /dev/fb0:/dev/fb0 <IMAGE_ID>
```

## Conclusione

Ora si ha un'applicazione LVGL in esecuzione. Come procedere?

- Ora si è pronti per compilare l'applicazione LVGL per Torizon OS. Si consiglia di familiarizzare con l'estensione IDE VSCode, poiché semplificherà il flusso di lavoro.
- VSCode, è il modo migliore per sviluppare per Torizon OS. Se si usa un altro editor o IDE, si può sempre scrivere script per automatizzare le operazioni di compilazione/pushing/pulling.
- Leggere questo [articolo](#) e capire come progettare applicazioni per Torizon OS.
- Si consiglia inoltre di familiarizzare con Torizon Cloud e Torizon OS Builder quando ci si avvicina alla produzione.

## 4.9.9 Yocto

The Yocto Project (YP) è un progetto collaborativo open source che aiuta gli sviluppatori a creare sistemi Linux personalizzati, indipendentemente dall'architettura hardware.

Il progetto fornisce un set flessibile di strumenti e uno spazio in cui gli sviluppatori embedded di tutto il mondo possono condividere tecnologie, stack software, configurazioni e best practice, utilizzabili per creare immagini Linux personalizzate per dispositivi embedded e IoT, o ovunque sia necessario un sistema operativo Linux personalizzato.

L'obiettivo di questa sezione è facilitare la comprensione dei concetti di base di Yocto e aiutare i principianti a iniziare a utilizzarlo.

### Componenti principali del progetto Yocto

L'esecutore di task BitBake, insieme a vari tipi di file di configurazione, costituisce l'OpenEmbedded-Core (OE-Core). Questa sezione fornisce una panoramica di questi componenti, descrivendone l'utilizzo e le interazioni.

BitBake gestisce l'analisi e l'esecuzione dei file di dati. I dati stessi sono di vario tipo:

- *Ricette*: Forniscono dettagli su particolari componenti software.
- *Dati di Classi*: Estraggono informazioni di build comuni (ad esempio, come compilare un kernel Linux).
- *Dati di Configurazione*: Definiscono impostazioni specifiche della macchina, decisioni sulle policy e così via. I dati di configurazione fungono da collante per collegare tutto

BitBake sa come combinare più sorgenti di dati e fa riferimento a ciascuna sorgente dati come a un layer.

Ecco alcuni brevi dettagli su questi componenti principali.

### BitBake

BitBake è lo strumento al centro del sistema di compilazione OpenEmbedded ed è responsabile dell'analisi dei metadati, della generazione di un elenco di task da essi e della loro esecuzione.

Questa sezione introduce brevemente BitBake. Per maggiori informazioni su BitBake, consultare il [Manuale Utente di BitBake](#).

Per visualizzare un elenco delle opzioni supportate da BitBake, utilizzare uno dei seguenti comandi:

```
$ bitbake -h
$ bitbake --help
```

L'uso più comune di BitBake è `bitbake recipename`, dove `recipename` è il nome della ricetta che si desidera compilare (definito come "target"). Il target spesso corrisponde alla prima parte del nome file di una ricetta (ad esempio, "foo" per una ricetta denominata `foo_1.3.0-r0.bb`). Quindi, per elaborare il file della ricetta `matchbox-desktop_1.2.3.bb`, si potrebbe digitare quanto segue:

```
$ bitbake matchbox-desktop
```

Potrebbero esistere diverse versioni di `matchbox-desktop`. BitBake sceglie quella selezionata dalla configurazione della distribuzione. Per maggiori dettagli su come BitBake sceglie tra diverse versioni target e provider, consultare la sezione "[Preferences](#)" del Manuale Utente di BitBake.

BitBake tenta inoltre di eseguire prima eventuali attività dipendenti. Quindi, ad esempio, prima di compilare `matchbox-desktop`, BitBake compilerebbe un cross-compilatore e `glibc` se non fossero già stati compilati.

Un'opzione BitBake utile da considerare è `-k` o `--continue`. Questa opzione indica a BitBake di provare a continuare l'elaborazione del job il più a lungo possibile anche dopo aver riscontrato un errore. Quando si verifica un errore, il target che ha generato l'errore e quelli che dipendono da esso non possono essere ripristinati. Tuttavia, utilizzando questa opzione, è comunque possibile elaborare altre dipendenze.

## Ricette

I file con suffisso `.bb` sono file "ricette" (recipe). In generale, una ricetta contiene informazioni su un singolo software. Queste informazioni includono la posizione da cui scaricare il codice sorgente non modificato, eventuali patch da applicare a tale codice sorgente (se necessario), quali opzioni di configurazione speciali applicare, come compilare i file sorgente e come pacchettizzare l'output compilato.

Il termine "package" [pacchetto] viene talvolta utilizzato per riferirsi alle ricette. Tuttavia, poiché il termine "package" viene utilizzato per l'output pacchettizzato dal sistema di build di OpenEmbedded (ad esempio, i file `.ipk` o `.deb`), questo documento evita di utilizzare il termine "package" quando si fa riferimento alle ricette.

## Classi

I file di classe (`.bbclass`) contengono informazioni utili da condividere tra i file di ricette. Un esempio è la classe `autotools`\*, che contiene impostazioni comuni per qualsiasi applicazione compilata con `GNU Auto-tools` <[https://en.wikipedia.org/wiki/GNU\\_Autotools](https://en.wikipedia.org/wiki/GNU_Autotools)>. Il capitolo "[Classi](#)" nel Manuale di Riferimento del Progetto Yocto fornisce dettagli sulle classi e su come utilizzarle.

## Configurazioni

I file di configurazione (`.conf`) definiscono diverse variabili di configurazione che governano il processo di compilazione di OpenEmbedded. Questi file rientrano in diverse aree che definiscono le opzioni di configurazione della macchina, le opzioni di configurazione della distribuzione, le opzioni di ottimizzazione del compilatore, le opzioni di configurazione comuni generali e le opzioni di configurazione utente in `conf/local.conf`, che si trova nella Build Directory.

## I Layer

I layer sono repository che contengono metadati correlati (ovvero set di istruzioni) che indicano al sistema di build di OpenEmbedded come compilare un target. Il modello a layer del progetto Yocto facilita la collaborazione, la condivisione, la personalizzazione e il riutilizzo all'interno dell'ambiente di sviluppo Yocto Project. I layer separano logicamente le informazioni per il progetto. Ad esempio, è possibile utilizzare un layer per contenere tutte le configurazioni di un particolare componente hardware. Isolare le configurazioni specifiche dell'hardware consente di condividere altri metadati utilizzando un layer diverso, in cui tali metadati potrebbero essere comuni a diversi componenti hardware.

Nell'ambiente di sviluppo Yocto Project sono presenti molti layer. Sia [Yocto Project Compatible Layer Index](#) che [OpenEmbedded Layer Index](#) contengono layer che è possibile utilizzare o sfruttare.

Per convenzione, i layer nel Progetto Yocto seguono una forma specifica. Conformarsi a una struttura nota consente a BitBake di fare ipotesi durante le build su dove trovare i tipi di metadati. È possibile trovare procedure e informazioni

sugli strumenti (ad esempio **bitbake-layers**) per la creazione di layer adatti al Progetto Yocto nella sezione "comprensione e creazione di layer" dello Yocto Project Development Tasks Manual.

## LVGL in Yocto

Questo capitolo serve come guida per aiutare a creare una ricetta per un'applicazione che utilizza LVGL. Sebbene il processo verrà illustrato per Raspberry Pi 3 (64 bit), può essere applicato a qualsiasi board in modo simile.

### Pacchetti Host di Build

È necessario installare i pacchetti host essenziali sul proprio host di build. Il seguente comando installa i pacchetti host basati su una distribuzione Ubuntu

```
sudo apt install gawk wget git diffstat unzip texinfo gcc build-essential \
chrpath socat cpio python3 python3-pip python3-pexpect xz-utils \
debianutils iputils-ping python3-git python3-jinja2 python3-subunit zstd \
liblz4-tool file locales libacl1
```

#### Nota

Per i requisiti dei pacchetti host su tutte le distribuzioni Linux supportate, consultare la sezione [Required Packages for the Build Host](#) nello Yocto Project Reference Manual.

### Utilizzare Git per clonare i repository richiesti

Dopo questa sezione, l'albero delle cartelle sarà simile a questo

```
lvgl_yocto_guide/
└── build/
    ├── cache/
    ├── conf/
    ├── downloads/
    └── ...
    └── sources/
        ├── meta-openembedded
        ├── meta-raspberrypi
        └── poky
```

Iniziare la creazione dell'architettura delle cartelle

```
mkdir -p lvgl_yocto_guide/sources
```

Clonaare Poky, meta-openembedded e meta-raspberrypi nei sorgenti

```
cd lvgl_yocto_guide/sources
git clone --branch scarthgap https://git.yoctoproject.org/poky.git
git clone --branch scarthgap https://git.openembedded.org/meta-openembedded
git clone --branch scarthgap git://git.yoctoproject.org/meta-raspberrypi
```

### Creare un'immagine di base

Per capire meglio cosa sta succedendo, creiamo l'immagine come se fosse stata preparata per noi

```
cd ../../ # go back to the root folder
source sources/poky/oe-init-build-env
```

Ora si dovrebbe avere una cartella chiamata **build** accanto a **sources**. L'ultimo comando imposta anche la directory corrente sulla directory di build.

Nella directory di build, c'è una cartella **conf** con alcuni file al suo interno

```
└── build/
    └── conf/
        ├── bblayers.conf
        ├── conf-notes.txt
        ├── conf-summary.txt
        ├── local.conf
        └── templateconf.cfg
```

I file importanti per noi sono **local.conf** e **bblayers.conf**

Per aggiungere layer al progetto ci sono 2 opzioni:

### Comandi Bitbake (Opzione 1)

Si deve stare nello stesso terminale in cui è stato eseguito il comando `source`. Si aggiungolo i layer col comando Bitbake in questo modo:

```
bitbake-layers add-layer ../sources/meta-openembedded
bitbake-layers add-layer ../sources/meta-raspberrypi
```

### Modifica del file di configurazione (Opzione 2)

Aprire il file `conf/bblayers.conf` e aggiungere manualmente i path:

```
# POKY_BBLAYERS_CONF_VERSION is increased each time build/conf/bblayers.conf
# changes incompatibly
POKY_BBLAYERS_CONF_VERSION = "2"

BBPATH = "${TOPDIR}"
BBFILES ?= ""

BBLAYERS ?= " \
${TOPDIR}/../sources/poky/meta \
${TOPDIR}/../sources/poky/meta-poky \
${TOPDIR}/../sources/poky/meta-yocto-bsp \
${TOPDIR}/../sources/meta-raspberrypi \
${TOPDIR}/../sources/meta-openembedded/meta-oe \
${TOPDIR}/../sources/meta-openembedded/meta-multimedia \
${TOPDIR}/../sources/meta-openembedded/meta-networking \
${TOPDIR}/../sources/meta-openembedded/meta-python \
"
```

Per assicurarsi che i layer siano stati aggiunti come previsto, eseguire il comando Bitbake per visualizzarli tutti:

```
bitbake-layers show-layers
```

Dovrebbero essere elencati i seguenti layer:

- core
- yocto
- yoctobsp
- raspberrypi
- openembedded-layer
- multimedia-layer
- networking-layer
- meta-python

### Build per RaspberryPi3 64

Le configurazioni macchina disponibili per RaspberryPi possono essere elencate in questo modo:

```
ls ../sources/meta-raspberrypi/conf/machine/*.conf
```

Per compilare un'immagine per RaspberryPi3 64 bit, modificare il file `local.conf` sostituendo il valore di default `MACHINE ??=` in questo modo

```
MACHINE ??= "raspberrypi3-64"
```

Per compilare l'immagine da utilizzare, è necessario aggiungere anche quanto segue al file:

```
LICENSE_FLAGS_ACCEPTED = "synaptics-killswitch"
```

#### Nota

Ulteriori informazioni su questa variabile sono disponibili [qui](#).

Tutto è configurato, è ora di compilare l'immagine:

```
bitbake core-image-base
```

## Panoramica della build

Esaminiamo le cartelle di build per capire cosa è successo.

## Downloads

La posizione della cartella può essere modificata con la variabile `DL_DIR`.

La cartella Downloads è una parte fondamentale dell'ambiente di build di Yocto, in quanto funge da cache locale per tutte le sorgenti e le dipendenze esterne. Questo meccanismo di caching è essenziale per build efficienti, in quanto garantisce che build ripetute non comportino perdite di tempo nel download ripetuto degli stessi file.

## tmp/deploy

Questa cartella contiene gli artefatti di output finali pronti per il "deployment" [distribuzione]. Questi artefatti includono immagini del kernel, file system di root, pacchetti, SDK, file del bootloader e altri componenti generati durante il processo di build di Yocto.

Ci sono alcune cartelle chiave:

- **images**: it contains the images that can be flashed or deployed to the target device. Files like the Linux kernel, root filesystem (e.g., .ext4, .tar.gz, .squashfs), bootloaders (e.g., U-Boot), and other bootable images for the device are found here. It's organized by the machine (or target board) for which the image was built.
- **rmp/deb/ipk**: Queste cartelle contengono i singoli pacchetti software generati durante la compilazione, nel formato di pacchetto specificato (RPM, DEB o IPK). Questi pacchetti vengono in genere creati durante la compilazione del progetto Yocto con il supporto per la gestione dei pacchetti abilitato. Possono essere successivamente installati sul dispositivo di destinazione utilizzando gli strumenti di gestione dei pacchetti.
- **sdk**: Questa sottodirectory contiene i Software Development Kit (SDK) che è possibile utilizzare per la cross-compilazione delle applicazioni per il sistema di target. Gli SDK vengono generati quando si utilizza il comando `bitbake -c populate_sdk`.

## tmp/sysroots-components

Questa cartella rappresenta un approccio modulare introdotto in Yocto per la gestione delle sysroot. Suddivide la sysroot in singoli componenti, che possono essere considerati come blocchi costitutivi o pacchetti che costituiscono l'intero ambiente sysroot.

Ogni componente corrisponde a un pacchetto o dipendenza specifica che viene inserita nella sysroot. Questo approccio consente un riutilizzo più efficiente delle sysroot su più pacchetti o build e facilita il monitoraggio e la gestione delle dipendenze durante il processo di build.

La directory sysroot-components aiuta a ottimizzare il processo di build perché, se un pacchetto cambia, Yocto deve solo aggiornare il componente corrispondente anziché ricostruire o copiare l'intera sysroot.

Se sono stati seguiti i passaggi precedenti, ecco le cartelle che si troveranno:

- **all**: File indipendenti dall'architettura.
- **cortexa53**: File per l'architettura Cortex-A53 (ARMv8-A).
- **manifests**: File "track" installati nella sysroot per pacchetto.
- **raspberrypi3\_64**: File specifici per Raspberry Pi 3 (64 bit).
- **x86\_64**: File per l'architettura x86\_64 (PC), in genere per tool di cross-compilazione.
- **x86\_64-nativesdk**: File relativi all'SDK per la cross-compilazione su un host x86\_64.

Ogni cartella corrisponde a componenti rilevanti per l'architettura specifica e, nel loro insieme, formano l'ambiente completo necessario per compilare ed eseguire il software per i sistemi target e host.

## tmp/sysroots-uninative

La directory sysroots-uninative in Yocto viene utilizzata per supportare la funzionalità "uninative", che consente build più riproducibili garantendo che l'ambiente di build rimanga coerente su diversi sistemi host. In sostanza, fornisce un modo per utilizzare gli stessi strumenti di build nativi su diverse distribuzioni Linux.

## tmp/work

La cartella `work` in Yocto è una directory chiave nella cartella `tmp` che contiene tutti gli artefatti di build temporanei per ogni pacchetto durante il processo di build. È qui che avviene la build e la compilazione effettiva dei singoli pacchetti o ricette. Ogni pacchetto (o "recipe" [ricetta]) elaborato da BitBake genera file e directory temporanei all'interno di questa cartella `work`.

La cartella `work` è in genere strutturata in base all'architettura della macchina e al pacchetto. Ecco come funziona in genere:

```
work/
└── <architecture>/
    └── <package>/
        └── <version>/
            ├── temp/
            ├── work/
            ├── sysroot-destdir/
            └── image/
```

Questa cartella è molto importante e di grande aiuto durante le fasi di debug. La sottocartella `temp` contiene log importanti che possono aiutare a diagnosticare i problemi di build. È qui che si andrà a cercare quando una build non riesce a capire cosa è andato storto.

## Ricetta LVGL

Maggiori informazioni sulle ricette sono disponibili nella sezione [Ricette](#).

C'è una ricetta in `meta-openembedded` fin dalla versione `honister`.

Tabella 3: Versione della ricetta lvgl

Branch	Ricetta
scarthgap (Yocto Project 5.0)	lvgl 9.1.0
nanbield (Yocto Project 4.3)	lvgl 8.3.10
mickledore (Yocto Project 4.2)	lvgl 8.1.0
langdale (Yocto Project 4.1)	lvgl 8.1.0
langdale (Yocto Project 4.1)	lvgl 8.1.0
kirkstone (Yocto Project 4.0)	lvgl 8.0.3

In questa guida, ci troviamo sul branch `scarthgap`, quindi utilizziamo lvgl 9.1.0.

Approfondiamo questa ricetta per capire cosa viene fatto. L'obiettivo è aggiungere questa libreria come oggetto "shared" nel rootfs target e generare un SDK con lvgl.

Questo è il path delle ricette lvgl: `lvgl_yocto_guide/sources/meta-openembedded/meta-oe/recipes-graphics/lvgl`

Ecco l'architettura della cartella delle ricette lvgl:

```
lvgl
├── files
│   ├── 0002-fix-sdl-handle-both-LV_IMAGE_SRC_FILE-and-LV_IMAGE_S.patch
│   ├── 0003-Make-fbdev-device-node-runtime-configurable-via-envi.patch
│   ├── 0004-Factor-out-fbdev-initialization-code.patch
│   ├── 0005-Add-DRM-KMS-example-support.patch
│   ├── 0006-Add-SDL2-example-support.patch
│   ├── 0007-fix-cmake-generate-versioned-shared-libraries.patch
│   └── 0008-fix-fbdev-set-resolution-prior-to-buffer.patch
└── lv-conf.inc
    └── lvgl_9.1.0.bb
        └── lvgl-demo-fb_9.1.0.bb
```

- La cartella `file` contiene tutte le patch che possono essere applicate durante la compilazione della ricetta.

- `lv_conf.inc` è un file include, che di solito contiene impostazioni di configurazione comuni per LVGL che possono essere condivise tra più ricette.
- `lvgl_9.1.0.bb` è la ricetta per compilare la libreria lvgl.
- `lvgl-demo-fb_9.1.0.bb` è una ricetta per compilare un'applicazione utilizzando lvgl.

Per ora, comprendiamo la ricetta della libreria lvgl.

```
# SPDX-FileCopyrightText: Huawei Inc.
#
# SPDX-License-Identifier: MIT

HOMEPAGE = "https://lvgl.io/"
DESCRIPTION = "LVGL is an OSS graphics library to create embedded GUI"
SUMMARY = "Light and Versatile Graphics Library"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://LICENCE.txt;md5=b1198c89ae87f043108cea62460b03a"

SRC_URI = "\n    git://github.com/lvgl/lvgl;protocol=https;branch=master \
    file://0002-fix-sdl-handle-both-LV_IMAGE_SRC_FILE-and-LV_IMAGE_S.patch \
    file://0007-fix-cmake-generate-versioned-shared-libraries.patch \
    file://0008-fix-fbdev-set-resolution-prior-to-buffer.patch \
"
SRCREV = "e1c0b21b2723d391b885de4b2ee5cc997eccca91"

inherit cmake

EXTRA_OECMAKE = "-DLIB_INSTALL_DIR=${baselib} -DBUILD_SHARED_LIBS=ON"
S = "${WORKDIR}/git"

require lv-conf.inc

do_install:append() {
    install -d "${D}/${includedir}/${PN}"
    install -m 0644 "${S}/lv_conf.h" "${D}/${includedir}/${PN}/lv_conf.h"
}

FILESTOINSTALL += "\n    ${includedir}/${PN}/ \
"
```

**LICENSE:** Indica la licenza del software, indicando che è distribuito con licenza MIT.

**LIC\_FILES\_CHKSUM:** Questa variabile contiene un checksum (in questo caso, un hash MD5) per il file di licenza (qui si presume sia LICENSE.txt). Questo aiuta a garantire l'integrità del file di licenza incluso nel pacchetto. Nel repository LVGL, è presente un file LICENSE.txt. Per ottenere il valore dell'hash del file, è possibile eseguire questo comando: `md5sum LICENSE.txt`

**SRC\_URI:** Specifica le posizioni del codice sorgente e delle patch per la ricetta:

- Il repository sorgente principale per LVGL che specifica il branch master.
- I seguenti argomenti sono i file di patch locali che verranno applicati al codice sorgente durante il processo di build.

**SRCPREV:** Definisce il commit specifico (in questo caso, un hash Git SHA) da cui verrà recuperato il codice sorgente. Questo garantisce che la build utilizzi una versione coerente del codice.

**inherit:** Questa riga indica che la ricetta utilizza la classe `cmake`, che fornisce funzionalità per la creazione di progetti che utilizzano CMake come sistema di build.

**EXTRA\_OECMAKE:** Opzioni aggiuntive passate a CMake durante la fase di configurazione.

**S:** Questa variabile definisce la directory sorgente in cui verrà posizionato il codice sorgente decompresso dopo il recupero.  `${WORKDIR}` è una variabile standard in BitBake che punta alla directory di lavoro per la ricetta.  
**require:** Questa riga include un altro file di configurazione, `lv-conf.inc`, che probabilmente contiene opzioni di configurazione aggiuntive o variabili specifiche della libreria LVGL.

**FILES:** Questa è una variabile BitBake utilizzata per specificare i file che devono essere inclusi in un particolare pacchetto. In questo caso, la variabile specifica i file per un pacchetto correlato allo sviluppo (ad esempio, file header).

## I Task della Ricetta

Quando una ricetta viene compilata, eseguirà più task. È possibile eseguire ogni task manualmente per capire cosa viene generato in ogni passaggio, oppure è possibile eseguire `bitbake lvgl` per eseguire tutti i task.

Fetch (do\_fetch)

```
bitbake lvgl -c fetch
```

Il task "fetch" recupera il codice sorgente del pacchetto dal repository locale o remoto.

L'indirizzo del Repository di recupero dev'essere memorizzato nella variabile **SRC\_URI**. Nella variabile **SRCREV** viene definito l'hash di commit del repository GitHub.

Una volta completato il task di recupero (fetch), è possibile trovare i codici sorgente recuperati in **build/downloads**.

Per questa ricetta, si troverà una nuova cartella qui: **lvgl\_yocto\_guide build/downloads/git2/github.com.lvgl.lvgl**.

Si può anche trovare la cartella architecture creata in **lvgl\_yocto\_guide/ build/tmp/work/cortexa53-poky-linux/lvgl** ma queste cartelle sono vuote poiché è stato eseguito solo il fetch.

#### Unpack (do\_unpack)

```
bitbake lvgl -c unpack
```

Il task "unpack" scompatta il pacchetto scaricato co task Fetch.

Nella cartella **lvgl\_yocto\_guide/build/tmp/work/cortexa53-poky-linux/lvgl/9.1.0**, ora si può trovare il codice sorgente in **git** (come definito nella ricetta). Si vedranno anche le patch che verranno applicate nel passaggio successivo. Quindi, per ora, i sorgenti non sono modificati e sono uguali al commit specificato.

#### Patch (do\_patch)

```
bitbake lvgl -c patch
```

Il task "patch" individua i file delle patch e applica le patch ai sorgenti, se disponibili. Questo è un task facoltativo, eseguito se la patch è disponibile.

Il file della patch è definito anche nella variabile **SRC\_URI**. Per default viene eseguito nella directory sorgente corrente  **\${S}**.

#### Configure (do\_configure)

```
bitbake lvgl -c configure
```

Il task di configurazione configura il codice sorgente abilitando e disabilitando eventuali opzioni di build e di configurazione per il software in fase di compilazione prima della compilazione, se disponibile.

Questo è un passaggio facoltativo, eseguito se disponibile.

In questo caso, crea una directory di build e richiama CMake per configurare il progetto, specificando le opzioni di build e i path in base alle variabili nella ricetta. Genera i Makefile o i file di progetto necessari per la build. Inoltre, ci sono operazioni aggiunte al task in **lv-conf.inc**.

Quindi, alla fine del task, in **lvgl\_yocto\_guide/build/tmp/work/cortexa53-poky-linux/lvgl/9.1.0**, troverete una cartella **build** che è stata generata eseguendo il comando CMake, ma non è ancora stato compilato nulla. Inoltre, le sysroot contengono tutto il necessario per compilare la libreria lvgl.

#### Compile (do\_compile)

```
bitbake lvgl -c compile
```

Il task Compilation compila il codice sorgente se sono disponibili passaggi di compilazione e genera un file binario.

Questo è un passaggio facoltativo, eseguito se la compilazione è disponibile.

Se sono presenti passaggi di compilazione, questi vengono definiti nella funzione **do\_compile()** di bitbake.

Come nel task precedente, questa operazione viene gestita da **inherit cmake**.

Nella cartella di build, ora è possibile visualizzare la libreria compilata. I file .so sono disponibili in `lvgl_yocto_guide/build/tmp/work/ cortexa53-poky-linux/lvgl/9.1.0/build/lib`.

Una volta completata questa attività, tutto è pronto per l'installazione.

#### Install (do\_install)

```
bitbake lvgl -c install
```

Il task Install copia i file da impacchettare nell'area di archiviazione \${D}. Questo viene eseguito con la directory di lavoro corrente \${S}, che è la directory di compilazione.

Crea la directory necessaria per i file header nella directory di installazione di destinazione. Installa il file header `lv_conf.h` dalla directory sorgente in quella di inclusione appropriata nel path di destinazione, assicurandosi che abbia i permessi corretti.

Anche la libreria lvgl (file .so) è pronta per essere installata nell'immagine finale. È stata creata una nuova cartella `image`.

#### ● Importante

Nel file `build/conf/local.conf`, aggiungere queste 2 righe alla fine del file:

```
IMAGE_INSTALL:append = " lvgl"
TOOLCHAIN_HOST_TASK:append = " lvgl"
```

Questo aggiungerà la libreria lvgl all'immagine generata e la aggiungerà anche all'SDK host che genereremo in seguito.

Con queste modifiche, ora si può eseguire nuovamente la ricetta dell'immagine:

```
bitbake core-image-base
```

Questo eseguirà tutte le attività descritte in precedenza.

Se tutto è andato bene, si troverà il file `build/tmp/deploy/rpm/cortexa53/lvgl-9.1.0-r0.cortexa53.rpm` e altri file rpm relativi a lvgl.

## Generazione dell'SDK

La generazione di un Software Development Kit (SDK) in Yocto svolge diverse funzioni importanti, in particolare nello sviluppo embedded:

- Ambiente di sviluppo:** L'SDK fornisce agli sviluppatori un ambiente di sviluppo pronto all'uso, personalizzato per una specifica piattaforma hardware di destinazione. Questo include gli strumenti, le librerie e gli header necessari per creare applicazioni eseguibili sul dispositivo target.
- Cross-Compilazione:** L'SDK consente agli sviluppatori di compilare in modo incrociato le applicazioni da una macchina host (tipicamente con architettura x86 o x86\_64) per un'architettura diversa (ad esempio, ARM). Questo è essenziale per i sistemi embedded, dove lo sviluppo avviene spesso su macchine più potenti.
- Sviluppo Semplificato:** Impacchettando tutti i componenti necessari, l'SDK semplifica il processo di configurazione di un ambiente di sviluppo. Gli sviluppatori non devono installare e configurare manualmente strumenti e librerie; tutto il necessario è incluso nell'SDK.
- Ambiente di Build Coerente:** L'SDK garantisce che gli sviluppatori lavorino con le stesse versioni e gli stessi strumenti utilizzati nella build di Yocto, il che aiuta a evitare problemi di compatibilità e garantisce che le applicazioni si comportino come previsto sul dispositivo target.

Per generare un SDK dell'ambiente, eseguire il seguente comando:

```
bitbake core-image-base -c populate_sdk
```

This will create a script `build/tmp/deploy/sdk/poky-glibc-x86_64-core-image-base-cortexa53-raspber0.4.sh`. This script allows you to install the SDK where you prefer. Here is the execution output of the script

```
$ ./sdk/poky-glibc-x86_64-core-image-base-cortexa53-raspberrypi3-64-toolchain-5.0.4.sh
Poky (Yocto Project Reference Distro) SDK installer version 5.0.4
=====
Enter target directory for SDK (default: /opt/poky/5.0.4): /opt/poky/sdk-with-lvgl
You are about to install the SDK to "/opt/poky/sdk-with-lvgl". Proceed [Y/n]? y
```

Per assicurarsi che l'SDK sia stato generato con lvgl installato, si va al path in cui è stato estratto l'SDK e si trovano tutti i file lvgl:

```
cd /opt/poky/5.0.4/sysroots/cortexa53-poky-linux
find . -name "*lvgl*"
```

I file . so che si troveranno dipenderanno dalla configurazione LVGL utilizzata.

Ora, per utilizzare l'ambiente SDK e cross-compilare un'applicazione:

```
source /opt/poky/5.0.4/environment-setup-cortexa53-poky-linux
```

### Nota

I tool disponibili nella libreria LVGL dipenderanno dalla configurazione di build con cui è stata realizzata la ricetta.

## Ricetta personalizzata

Fino a questa sezione, tutto era già stato preparato. Abbiamo utilizzato ricette esistenti. L'obiettivo qui è creare una ricetta da zero e aggiungere il binario generato all'immagine.

### Creare un layer

Per prima cosa, si crea un layer e lo si aggiunge al file di configurazione.

```
bitbake-layers create-layer ../sources/meta-myvg
bitbake-layers add-layer ../sources/meta-myvg
```

Nella cartella **sources**, è stata creata una nuova cartella: **meta-myvg** e l'albero delle directory dovrebbe apparire come segue.

```
|- conf
  |- layer.conf
|- COPYING.MIT
|- README
|- recipes-example
  |- example
    |- example_0.1.bb
```

### Creare una ricetta

Seguendo questa struttura, si crea una cartella contenente le ricette per compilare una o più applicazioni utilizzando lvgl.

```
cd ../sources/meta-myvg
mkdir -p recipes-lvglapp/lvgl-fbdev-benchmark/files
touch recipes-lvglapp/lvgl-fbdev-benchmark/lvglbenchmarkfbdev_2.4.bb
```

Ci concentreremo su un'applicazione che clonerà un repository git lvgl e lo patcherà per le nostre esigenze.

The content of **recipes-lvglapp/lvgl-fbdev-benchmark/lvglbenchmarkfbdev\_2.4.bb**

```
DESCRIPTION = "My C++ lvgl app with CMake"
LICENSE = "MIT"
LIC_FILES_CHKSUM = "file://${COMMON_LICENSE_DIR}/MIT;md5=0835ade698e0bcf8506ecda2f7b4f302"

# Update SRC_URI to point to the GitHub repository
SRC_URI = "git://github.com/lvgl/lv_port_linux.git;branch=master;protocol=https \
           file://0001-change-config-file-to-run-fbdev-with-2-threads.patch \
           file://0002-adapt-CMakeLists-file-to-compile-and-link-fbdev.patch \
           "

S = "${WORKDIR}/git"
inherit cmake
```

(continues on next page)

(continua dalla pagina precedente)

```
CMAKE_PROJECT_NAME = "lvgl_app"
CMAKE_PROJECT_VERSION = "2.0"

do_install() {
    install -d ${D}${bindir}
    install -m 0755 ${S}/bin/lvglbenchmark ${D}${bindir}
}
```

I sorgenti provengono dal repository `lv_port_linux`. Applichiamo 2 patch per modificare `CMakeLists.txt` e `lv_conf.h`.

## Patch 1

Creazione del primo file patch

```
touch 0001-change-config-file-to-run-fbdev-with-2-threads.patch
```

Contenuto di `0001-change-config-file-to-run-fbdev-with-2-threads.patch`

```
From d49d475d57f69d6172a6b38095ebf3b887f0e731 Mon Sep 17 00:00:00 2001
From: Gabriel Catel Torres <gabriel.catel@edgemtech.ch>
Date: Tue, 1 Oct 2024 10:28:55 +0200
Subject: [PATCH] change config file to run fbdev with 2 threads

...
lv_conf.h | 34 ++++++----- 1 file changed, 17 insertions(+), 17 deletions(-)

diff --git a/lv_conf.h b/lv_conf.h
index 62a834f..58fbe7a 100644
--- a/lv_conf.h
+++ b/lv_conf.h
@@ -39,9 +39,9 @@
 * - LV_STDLIB_RTTHREAD: RT-Thread implementation
 * - LV_STDLIB_CUSTOM: Implement the functions externally
 */
#define LV_USE_STDLIB_MALLOC LV_STDLIB_BUILTIN
#define LV_USE_STDLIB_STRING LV_STDLIB_BUILTIN
#define LV_USE_STDLIB_SPRINTF LV_STDLIB_BUILTIN
#define LV_USE_STDLIB_MALLOC LV_STDLIB_CLIB
#define LV_USE_STDLIB_STRING LV_STDLIB_CLIB
#define LV_USE_STDLIB_SPRINTF LV_STDLIB_CLIB

#define LV_STDINT_INCLUDE <stdint.h>
#define LV_STDOF_INCLUDE <stddef.h>
@@ -89,7 +89,7 @@
 * - LV_OS_WINDOWS
 * - LV_OS_MQX
 * - LV_OS_CUSTOM */
#define LV_USE_OS LV_OS_NONE
#define LV_USE_OS LV_OS_PTHREAD

#if LV_USE_OS == LV_OS_CUSTOM
#define LV_OS_CUSTOM_INCLUDE <stdint.h>
@@ -117,12 +117,12 @@
 * and can't be drawn in chunks. */

/* The target buffer size for simple layer chunks. */
#define LV_DRAW_LAYER_SIMPLE_BUF_SIZE (24 * 1024) /* [bytes] */
#define LV_DRAW_LAYER_SIMPLE_BUF_SIZE (512 * 1024) /* [bytes] */

/* The stack size of the drawing thread.
 * NOTE: If FreeType or ThorVG is enabled, it is recommended to set it to 32KB or more.
 */
#define LV_DRAW_THREAD_STACK_SIZE (8 * 1024) /* [bytes] */
#define LV_DRAW_THREAD_STACK_SIZE (32 * 1024) /* [bytes] */

#define LV_USE_DRAW_SW
#if LV_USE_DRAW_SW == 1
@@ -147,7 +147,7 @@
 /* Set the number of draw unit.
 * > 1 requires an operating system enabled in 'LV_USE_OS'
 * > 1 means multiple threads will render the screen in parallel */
#define LV_DRAW_SW_DRAW_UNIT_CNT 1
#define LV_DRAW_SW_DRAW_UNIT_CNT 2

/* Use Arm-2D to accelerate the sw render */
#define LV_USE_DRAW_ARM2D_SYNC 0
@@ -979,8 +979,8 @@
#define LV_USE_LINUX_FBDEV 1
#if LV_USE_LINUX_FBDEV
#define LV_LINUX_FBDEV_BSD 0
-#define LV_LINUX_FBDEV_RENDER_MODE LV_DISPLAY_RENDER_MODE_PARTIAL
-#define LV_LINUX_FBDEV_BUFFER_COUNT 0
+#define LV_LINUX_FBDEV_RENDER_MODE LV_DISPLAY_RENDER_MODE_DIRECT
+#define LV_LINUX_FBDEV_BUFFER_COUNT 2
#define LV_LINUX_FBDEV_BUFFER_SIZE 60
#endif

@@ -1069,19 +1069,19 @@
#define LV_USE_DEMO_WIDGETS 1

/* Demonstrate the usage of encoder and keyboard */
#define LV_USE_DEMO_KEYPAD_AND_ENCODER 1
#define LV_USE_DEMO_KEYPAD_AND_ENCODER 0

/* Benchmark your system */
#define LV_USE_DEMO_BENCHMARK 1

/* Render test for each primitives. Requires at least 480x272 display */
#define LV_USE_DEMO_RENDER 1
#define LV_USE_DEMO_RENDER 0
```

(continues on next page)

(continua dalla pagina precedente)

```
/* Stress test for LVGL */
#define LV_USE_DEMO_STRESS 1
#define LV_USE_DEMO_STRESS 0

/* Music player demo */
#define LV_USE_DEMO_MUSIC 1
#define LV_USE_DEMO_MUSIC 0
#if LV_USE_DEMO_MUSIC
#define LV_DEMO_MUSIC_SQUARE 0
#define LV_DEMO_MUSIC_LANDSCAPE 0
@@ -1091,16 +1091,16 @@
#endif

/* Flex layout demo */
#define LV_USE_DEMO_FLEX_LAYOUT 1
#define LV_USE_DEMO_FLEX_LAYOUT 0

/* Smart-phone like multi-language demo */
#define LV_USE_DEMO_MULTILANG 1
#define LV_USE_DEMO_MULTILANG 0

/* Widget transformation demo */
#define LV_USE_DEMO_TRANSFORM 1
#define LV_USE_DEMO_TRANSFORM 0

/* Demonstrate scroll settings */
#define LV_USE_DEMO_SCROLL 1
#define LV_USE_DEMO_SCROLL 0

/* Vector graphic demo */
#define LV_USE_DEMO_VECTOR_GRAPHIC 0
--
```

2.34.1

## Patch 2

Creazione del primo file patch

```
touch 0002-adapt-CMakeLists-file-to-compile-and-link-fbdev.patch
```

Contenuto di 0002-adapt-CMakeLists-file-to-compile-and-link-fbdev.patch

```
From ad464e0683aced74085fb89895b8d040ecad0206 Mon Sep 17 00:00:00 2001
From: Gabriel Catel Torres <gabriel.catel@edgemtech.ch>
Date: Tue, 1 Oct 2024 10:31:29 +0200
Subject: [PATCH] adapt CMakeLists file to compile and link only for fbdev

---
CMakeLists.txt | 17 ++++++-----  

1 file changed, 5 insertions(+), 12 deletions(-)

diff --git a/CMakeLists.txt b/CMakeLists.txt  

index 658193f..ad56cc2 100644  

--- a/CMakeLists.txt  

+++ b/CMakeLists.txt  

@@ -1,8 +1,8 @@  

cmake_minimum_required(VERSION 3.10)  

project(lvgl)

-set(CMAKE_C_STANDARD 99) # lvgl officially support C99 and above
-set(CMAKE_CXX_STANDARD 17) #C17
+set(CMAKE_C_STANDARD 99)
+set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
@@ -10,15 +10,8 @@ set(EXECUTABLE_OUTPUT_PATH ${PROJECT_SOURCE_DIR}/bin)
add_subdirectory(lvgl)
target_include_directories(lvgl PUBLIC ${PROJECT_SOURCE_DIR})

-add_executable(main main.c mouse_cursor_icon.c)
+add_executable(lvglbenchmark main.c mouse_cursor_icon.c)

-include(${CMAKE_CURRENT_LIST_DIR}/lvgl/tests/FindLibDRM.cmake)
-include_directories(${Libdrm_INCLUDE_DIRS})
-
-find_package(SDL2)
-find_package(SDL2_image)
-include_directories(${SDL2_INCLUDE_DIRS} ${SDL2_IMAGE_INCLUDE_DIRS})
-
-target_link_libraries(main lvgl lvgl::examples lvgl::demos lvgl::thorvg ${SDL2_LIBRARIES} ${SDL2_IMAGE_LIBRARIES} ${Libdrm_LIBRARIES} m pthread)
-add_custom_target (run COMMAND ${EXECUTABLE_OUTPUT_PATH}/main DEPENDS main)
+target_link_libraries(lvglbenchmark lvgl lvgl::examples lvgl::demos lvgl::thorvg m pthread)
+add_custom_target (run COMMAND ${EXECUTABLE_OUTPUT_PATH}/lvglbenchmark DEPENDS main)
-
--
```

2.34.1

## Compilare la ricetta

Ora si dovrebbe essere in grado di visualizzare la ricetta con l'elenco delle ricette esistenti.

```
bitbake-layers show-recipes | grep lvglbenchmarkfbdev
```

Si aggiunge la ricetta al progetto. Modificare il file `build/conf/local.conf` aggiungendo questa riga.

```
IMAGE_INSTALL:append = " lvglbenchmarkfbdev"
```

Ora compilare l'immagine.

```
bitbake core-image-base
```

Eseguire questo comando per assicurarsi che il binario sia stato generato e installato nel rootfs.

```
# Run this command in build/tmp folder
find . -wholename "*bin/lvglbench"
```

## Terminologia dello Yocto Project

Iniziare a usare Yocto può essere complicato. Sono molti i termini utilizzati, specifici per Yocto e per l'ambiente Bitbake.

Un elenco di termini e definizioni che gli utenti alle prime armi con l'ambiente di sviluppo del progetto Yocto potrebbero trovare utili è disponibile [qui](#).

## Glossario delle Variabili di Yocto

Questo capitolo elenca le variabili di base utilizzate nella guida LVGL di Yocto e fornisce una panoramica della loro funzione e del loro contenuto.

A more complete variable glossary can be found in [Yocto Variable Glossary](#). This section covers a lot of variables used in the OpenEmbedded build system.

### S

Posizione nella "Build Directory" in cui risiede il codice sorgente della ricetta spacchettata. Per default, questa directory è \${WORKDIR}/\${BPN}-\${PV}, dove \${BPN} è il nome della ricetta di base e \${PV} è la versione della ricetta. Se il tarball sorgente estrae il codice in una directory con un nome diverso da \${BPN}-\${PV}, o se il codice sorgente viene recuperato da un SCM come Git o Subversion, è necessario impostare S nella ricetta in modo che il sistema di build di OpenEmbedded sappia dove trovare il codice sorgente decompresso.

Ad esempio, si supponga una "Source Directory" di primo livello denominata poky e una "Build Directory" di default in poky/build. In questo caso, la directory di lavoro utilizzata dal sistema di build per conservare la ricetta decompressa per db è la seguente:

```
poky/build/tmp/work/qemux86-poky-linux/db/5.1.19-r3/db-5.1.19
```

Il codice sorgente decompresso si trova nella cartella db-5.1.19.

Il prossimo esempio presuppone un repository Git. Per default, i repository Git vengono clonati in \${WORKDIR}/git during do\_fetch. Poiché questo path è diverso dal valore predefinito di S, è necessario impostarlo specificamente in modo che il sorgente possa essere individuato:

```
SRC_URI = "git://path/to/repo.git;branch=main"
S = "${WORKDIR}/git"
```

### D

Directory di destinazione. La posizione nella Build Directory in cui i componenti vengono installati dal task do\_install. Questa posizione di default è:

```
 ${WORKDIR}/image
```

#### Nota

I task che leggono o scrivono in questa directory devono essere eseguiti con fakeroott.

## B

La directory all'interno della Build Directory in cui il sistema di build OpenEmbedded inserisce gli oggetti generati durante il processo di build di una ricetta. Per default, questa directory è la stessa della directory S, definita come:

```
S = "${WORKDIR}/${BP}"
```

È possibile separare la directory (S) e la directory puntata dalla variabile B. La maggior parte delle ricette basate su Autotools supporta la separazione di queste directory. Il sistema di build utilizza per default directory separate per gcc e alcune ricette del kernel.

## WORKDIR

Il percorso della directory di lavoro in cui il sistema di build di OpenEmbedded crea una ricetta. Questa directory si trova all'interno della struttura di directory TMPDIR ed è specifica della ricetta in fase di compilazione e del sistema per cui viene compilata.

La directory WORKDIR è definita come segue:

```
${TMPDIR}/work/${MULTIMACH_TARGET_SYS}/${PN}/${EXTENDPE}${PV}-${PR}
```

La directory effettiva dipende da diversi fattori:

- **TMPDIR**: Directory di output della build di livello superiore
- **MULTIMACH\_TARGET\_SYS**: Identificativo del sistema target
- **PN**: Nome della ricetta
- **EXTENDPE**: Epoch: se PE non è specificato, come solitamente accade per la maggior parte delle ricette, EXTENDPE è vuoto.
- **PV**: Versione della ricetta
- **PR**: Revisione della ricetta

Ad esempio, supponiamo una Source Directory di livello superiore denominata poky, una Build Directory predefinita in poky/build e un sistema target qemux86-poky-linux. Inoltre, supponiamo che la ricetta si chiami foo\_1.3.0-r0.bb. In questo caso, la directory di lavoro utilizzata dal sistema di build per compilare il pacchetto sarebbe la seguente:

```
poky/build/tmp/work/qemux86-poky-linux/foo/1.3.0-r0
```

## PN

Questa variabile può avere due funzioni distinte a seconda del contesto: un nome di ricetta o un nome di pacchetto risultante.

PN si riferisce a un nome di ricetta nel contesto di un file utilizzato dal sistema di build di OpenEmbedded come input per creare un pacchetto. Il nome viene normalmente estratto dal nome del file della ricetta. Ad esempio, se la ricetta si chiama expat\_2.0.1.bb, il valore di default di PN sarà "expat".

La variabile si riferisce a un nome di pacchetto nel contesto di un file creato o prodotto dal sistema di build di OpenEmbedded.

Se applicabile, la variabile PN contiene anche eventuali suffissi o prefissi speciali. Ad esempio, utilizzando bash per compilare pacchetti per la macchina nativa, PN è bash-native. Utilizzando bash per compilare pacchetti per il target e per Multilib, PN sarebbe rispettivamente bash e lib64-bash.

## PR

La revisione della ricetta. Il valore di default per questa variabile è "r0". Le revisioni successive della ricetta hanno convenzionalmente i valori "r1", "r2" e così via. Quando il PV aumenta, il PR viene convenzionalmente reimpostato a "r0".

**i Nota**

Il sistema di build di OpenEmbedded non ha bisogno dell'ausilio del PR per sapere quando ri-buildare una ricetta. Il sistema di build utilizza i checksum di input dei task insieme ai meccanismi di "stamp" [timbro] e cache di stato condivisa.

La variabile PR diventa significativa principalmente quando un gestore di pacchetti installa dinamicamente i pacchetti su un'immagine già compilata. In questo caso, il PR, che è il valore di default di PKGR, aiuta il gestore di pacchetti a distinguere quale pacchetto sia il più recente nei casi in cui molti pacchetti abbiano lo stesso PV (ad esempio, PKGV). Un componente con molti pacchetti con lo stesso PV di solito significa che tutti i pacchetti installano la stessa versione upstream, ma con pacchetti di versioni successive (PR) che includono correzioni al packaging.

**i Nota**

Il PR non deve essere aumentato per le modifiche che non alterano il contenuto o i metadati del pacchetto.

Poiché la gestione manuale delle PR può essere macchinosa e soggetta a errori, esiste una soluzione automatizzata. Per ulteriori informazioni, consultare la sezione "[working with a pr service](#)" nello Yocto Project Development Tasks Manual.

**PV**

La versione della ricetta. La versione viene normalmente estratta dal nome del file della ricetta. Ad esempio, se la ricetta si chiama expat\_2.0.1.bb, il valore predefinito di PV sarà "2.0.1". PV generalmente non viene sovrascritto all'interno di una ricetta, a meno che non si stia compilando una versione instabile (ad esempio, di sviluppo) da un repository di codice sorgente (ad esempio Git o Subversion).

PV è il valore predefinito della variabile PKGV.

## 4.9.10 Zephyr

### Cos'è Zephyr?

Zephyr è un sistema operativo real-time (RTOS) [open source](#) facile da implementare, proteggere, connettere e gestire. Dispone di un set crescente di librerie software utilizzabili in diverse applicazioni e settori industriali, come l'IoT industriale, i dispositivi indossabili, il machine learning e altro ancora. Zephyr è progettato con particolare attenzione all'ampio supporto di chipset, alla sicurezza, all'affidabilità, alle release con supporto a lungo termine e a un ecosistema open source in crescita.

### Punti di Forza di Zephyr

- **Piccolo** - Funziona su microcontrollori con soli 8 kB di Flash e 5 kB di RAM.
- **Scalabile** - Utilizzabile per sistemi multicore complessi.
- **Personalizzabile** - Supporto immediato per oltre 500 schede e elevata portabilità.
- **Sicuro** - Progettato pensando alla sicurezza e alla protezione, offre supporto "Long-term" [a lungo termine].
- **Ecosistema** - Zephyr non fornisce solo il kernel RTOS, ma anche strumenti di sviluppo, device driver, connettività, logging, tracing, gestione dell'alimentazione e molto altro.
- **Disaccoppiamento** - Sfrutta il devicetree per descrivere e configurare il sistema target.
- **Conforme** - Le app sono eseguibili come applicazioni Linux native, semplificando il debug e la profilazione.

## Come eseguire LVGL su Zephyr?

Per configurare l'ambiente di sviluppo, consultare la [guida introduttiva](#).

Dopo aver completato la configurazione sopra descritta, si possono consultare tutti gli [esempi forniti](#) per diverse schede. Si può consultare l'elenco delle schede disponibili usando:

```
$ west boards
```

Dopo aver scelto una scheda, si può compilare una delle demo LVGL per essa. Qui utilizziamo la scheda `native_posix`, che consente di eseguire l'applicazione sul sistema host compatibile con Posix:

```
$ west build -b native_posix samples/modules/lvgl/demos
```

Per eseguire l'applicazione sul proprio host:

```
$ west build -t run
```

Se è stata scelta una qualsiasi delle altre schede supportate, si può flashare sul dispositivo con:

```
$ west flash
```

Per compilare una qualsiasi delle altre applicazioni demo, consultare gli esempi [README](#).

## Sfruttare le funzionalità di Zephyr

### Shell

Zephyr include una potente implementazione della shell abilitabile con i simboli Kconfig `CONFIG_SHELL` e `CONFIG_LV_Z_SHELL` (le demo precedenti la abilitano di default).

La shell offre la possibilità di abilitare/disabilitare i dispositivi LVGL:

```
# Create a new monkey with the given indev type
uart$ lvgl monkey create [pointer|keypad|button|encoder]

# Enable/Disable a monkey
uart$ lvgl monkey set <index> <inactive/active>
```

Questo è utile per verificare la presenza di "memory leak" [perdite di memoria] e altri bug nell'applicazione. A proposito di memory leak, è anche possibile acquisire statistiche sulla memoria utilizzata da LVGL.

```
uart$ lvgl stats memory
```

Per maggiori dettagli, consultare la documentazione della shell.

### Devicetree

Zephyr utilizza il linguaggio di descrizione devicetree per creare e gestire dispositivi di input LVGL.

Le descrizioni dei binding degli pseudo dispositivi sono disponibili all'indirizzo:

- `button` input
- `pointer` input
- `encoder` input
- `keypad` input

In sostanza, questi bufferizzano l'evento `input_event` generato dal dispositivo puntato dal phandle `input` oppure, se lasciato vuoto, il binding cattura tutti gli eventi indipendentemente dalla sorgente. Non è necessario istanziare o gestire i dispositivi manualmente, poiché vengono creati all'avvio dell'applicazione prima dell'esecuzione di `main()`.

La maggior parte delle schede o degli shield dotati di display o connettore per display hanno il dispositivo di input del puntatore già dichiarato:

```
lvgl_pointer {
    compatible = "zephyr,lvgl-pointer-input";
    input = <&ft5336_touch>;
};
```

È possibile accedere al `lv_indev_t` di lvgl sottostante per la configurazione. Esempio con il dispositivo encoder per assegnare un `lv_group_t`:

```
const struct device *lvgl_encoder = DEVICE_DT_GET(DT_COMPAT_GET_ANY_STATUS_OKAY(zephyr_lvgl_encoder_input));

lv_obj_t *arc;
lv_group_t *arc_group;

arc = lv_arc_create(lv_screen_active());
lv_obj_align(arc, LV_ALIGN_CENTER, 0, 0);
lv_obj_set_size(arc, 150, 150);

arc_group = lv_group_create();
lv_group_add_obj(arc_group, arc);
lv_indev_set_group(lvgl_input_get_indev(lvgl_encoder), arc_group);
```

## Kconfig

Oltre ad abilitare la shell, è possibile utilizzare Kconfig per ottimizzare il footprint dell'applicazione.

```
# Size of the memory region from which lvgl memory is allocated
CONFIG_LV_Z_MEM_POOL_SIZE=8192

# Do not include every widget/theme by default, enable them as needed.
CONFIG_LV_CONF_MINIMAL=y
```

Gli overlay possono essere utilizzati per abilitare/disabilitare funzionalità per schede specifiche o target di build. Per ulteriori informazioni, consultare la guida allo sviluppo di applicazioni.

## Ottimizzazione delle prestazioni in LVGL

Per ottimizzare le prestazioni di LVGL, è possibile configurare diverse opzioni kconfig:

- **CONFIG\_LV\_Z\_VDB\_SIZE**: Imposta la dimensione del buffer di rendering come percentuale dell'area di visualizzazione, regolabile dall'1% al 100%. Buffer più grandi possono migliorare le prestazioni, soprattutto se utilizzati con **CONFIG\_LV\_Z\_FULL\_REFRESH**.
- **CONFIG\_LV\_Z\_DOUBLE\_VDB**: Abilita l'utilizzo di due buffer di rendering, consentendo il rendering parallelo e il flushing dei dati, migliorando così la reattività e riducendo la latenza.
- **CONFIG\_LV\_Z\_VDB\_ALIGN**: Garantisce che il buffer di rendering sia correttamente allineato, il che è fondamentale per un accesso efficiente alla memoria in base alla profondità del colore.
- **CONFIG\_LV\_Z\_VBD\_CUSTOM\_SECTION**: Consente di posizionare i buffer di rendering in una sezione di memoria personalizzata (ad esempio, `.lvgl_buf`), utile per sfruttare tipi di memoria specifici come quella strettamente accoppiata o esterna per migliorare le prestazioni.

## Opzioni specifiche per Zephyr ≤ 3.7.0

Per le versioni di Zephyr 3.7.0 e precedenti, sono disponibili opzioni aggiuntive per gestire il flushing dei frame LVGL:

- **CONFIG\_LV\_Z\_FLUSH\_THREAD**: Abilita il flushing dei frame LVGL in un thread separato, consentendo al thread principale di continuare a eseguire il rendering del frame successivo simultaneamente. Questa opzione può essere disabilitata se non si desidera ottenere un miglioramento delle prestazioni.
  - **CONFIG\_LV\_Z\_FLUSH\_THREAD\_STACK\_SIZE**: Specifica la dimensione dello stack per il thread di flushing, con un valore di default di 1024 byte.
  - **CONFIG\_LV\_Z\_FLUSH\_THREAD\_PRIO**: Imposta la priorità del thread di flushing, con un valore predefinito di 0, che indica la priorità cooperativa.

Per le versioni più recenti di Zephyr, è possibile utilizzare OSAL (Operating System Abstraction Layer), che si occupa del flushing.

## Dove si possono trovare maggiori informazioni?

- Documentazione di Zephyr
- Mailing list di Zephyr
- Server Discord di Zephyr

## 4.10 Board

### 4.10.1 Toradex

#### Informazioni su Toradex

Toradex è specializzata nella produzione di soluzioni di elaborazione embedded di alta qualità basate su unità System-on-Module (SoM). Queste schede e moduli sono progettati per soddisfare le esigenze di applicazioni industriali e commerciali, offrendo affidabilità, longevità e scalabilità. Sfruttando framework standard di settore come lo Yocto Project, Toradex consente ai clienti di creare sistemi embedded personalizzati con facilità, garantendo compatibilità e flessibilità.

#### Portfolio Prodotti

Toradex offre un'ampia gamma di System-on-Module (SoM) e schede carrier progettate per diverse applicazioni e settori. Ecco una panoramica di base per aiutare a navigare nel portfolio.

#### Aquila

Aquila è un nuovo standard System on Module che espande il portfolio Toradex a nuovi livelli, offrendo prestazioni senza pari per carichi di lavoro CPU, machine learning e visione artificiale ad alta intensità di calcolo.

Per soddisfare i requisiti di intelligenza artificiale di nuova generazione in ambienti industriali e in altri ambienti difficili, Aquila introduce un connettore collaudato: il connettore board-to-board a 400 pin, che rende la scheda estremamente affidabile, soprattutto in questi ambienti.

Computer on Module Aquila:

- [Aquila iMX95](#)
- [Aquila AM69](#)

Schede Carrier Aquila:

- [Clover](#)
- [Aquila Development Board](#)

#### Apalis

Apalis è una famiglia scalabile di System on Module (SoM) / Computer on Module (CoM) che mira a fornire elevate prestazioni in un fattore di forma compatto. Apalis consente lo sviluppo di prodotti avanzati e robusti in tempi rapidi e con rischi ridotti. I moduli Apalis raggiungono le prestazioni più elevate nella gamma di prodotti Toradex e offrono il maggior numero di interfacce di qualsiasi altra linea di prodotti sui connettori MXM a 314 pin. I moduli Apalis sono adatti a un ampio numero di applicazioni in settori quali sanità, automazione industriale, robotica, smart city e molti altri. I moduli Apalis sono supportati dall'offerta software Toradex e dal ricco ecosistema di altri prodotti e servizi dell'azienda.

Computer on Module Apalis:

- [Apalis iMX8](#)
- [Apalis iMX6](#)
- [Apalis T30](#)

Schede Carrier Apalis:

- [Ixora](#)
- [Apalis Evaluation Board](#)

## Colibri

La famiglia The Colibri Arm® è composta da System on Module (SoM) / Computer on Module (CoM) di dimensioni SODIMM. La famiglia offre moduli miniaturizzati basati su NXP® i.MX 8X, i.MX 6ULL, i.MX 7, i.MX 6, Vybrid e NVIDIA® Tegra SoCs. Tutti questi moduli sono compatibili a livello di pin, garantendo la scalabilità delle applicazioni sviluppate su di essi. La famiglia Colibri offre progetti ottimizzati in termini di costi e prestazioni. È inoltre disponibile un'ampia gamma di schede carrier e accessori compatibili con Colibri, come fotocamere e display, per un facile sviluppo di prodotti sulla famiglia di moduli Colibri Arm.

Computer on Module Colibri:

- [Colibri iMX8X](#)
- [Colibri T30](#)
- [Colibri T20](#)
- [Colibri iMX6](#)
- [Colibri iMX7](#)
- [Colibri iMXiMX6ULL8X](#)
- [Colibri VF61](#)
- [Colibri VF50](#)

Schede Carrier Colibri:

- [Colibri Evaluation Board](#)
- [Iris Carrier Board](#)
- [Viola Carrier Board](#)
- [Aster Carrier Board](#)

## Verdin

Verdin è uno standard System on Module che amplia i fattori di forma di successo Colibri e Apalis. Offre un set di interfacce rivisitato, moderno e funzionale, puntando su facilità d'uso, robustezza e compatibilità con le esigenze future.

Verdin ha dimensioni simili a Colibri, ma con un numero di pin maggiore. Un connettore edge SODIMM DDR4 offre una connessione conveniente, altamente affidabile e resistente a urti e vibrazioni.

Computer on Module Verdin:

- [Verdin iMX8M Plus](#)
- [Verdin iMX8M Mini](#)
- [Verdin AM62](#)
- [i.MX 95 Verdin Evaluation Kit](#)
- [Verdin iMX95](#)

Schede Carrier Verdin:

- [Verdin Development Board con HDMI Adapter](#)
- [Dahlia Carrier Board con HDMI Adapter](#)
- [Yavia](#)
- [Mallow Carrier Board](#)
- [Ivy Carrier Board](#)

## TorizonOS

TorizonOS è un sistema operativo basato su Linux sviluppato da Toradex, pensato appositamente per i sistemi embedded. È progettato per semplificare lo sviluppo, l'implementazione e la manutenzione delle applicazioni, in particolare in ambito industriale e IoT. Combinando la facilità d'uso con potenti funzionalità come la containerizzazione e gli aggiornamenti over-the-air (OTA), TorizonOS aiuta gli sviluppatori ad accelerare il time-to-market, garantendo al contempo prestazioni e sicurezza elevate per le applicazioni embedded.

Una guida di TorizonOS per sviluppare un'applicazione utilizzando LVGL è disponibile nella sezione [Torizon OS](#).

Ulteriori informazioni sono disponibili nella documentazione di Torizon.

## Esempi di Toradex

Sono disponibili repository pronti all'uso. Cliccare [qui](#) per consultarli.

### 4.10.2 Riverdi

Riverdi è specializzata nella produzione di display di alta qualità e delle relative schede. Offrono soluzioni che spaziano dai semplici pannelli display ai display intelligenti, con tutto ciò che sta nel mezzo.

- Display embedded STM32
- Display per Computer Single Board
- Display intelligenti EVE
- Display HDMI
- Evaluation Board
- Display RGB, LVDS, MIPI DSI LCD

#### Display embedded STM32

I display embedded STM32 sono dotati di un MCU STM32 integrato che consente di integrare un'intera applicazione nel dispositivo. LVGL è particolarmente adatto a queste schede.

#### [lv\\_port\\_riverdi\\_stm32u5](#)

[lv\\_port\\_riverdi\\_stm32u5](#) è un porting del repository di LVGL che supporta tutti i Display Embedded Riverdi STM32 da 5 pollici.

- [RVT50HQSNWC00-B](#)
- [RVT50HQSNWC00](#)
- [RVT50HQSFWC0A](#)
- [RVT50HQSNWCA0](#)
- [RVT50HQSFWN00](#)
- [RVT50HQSNWN00](#)

Supporta nativamente la GPU NeoChrom integrata grazie al supporto LVGL per [Nema GFX](#). NemaVG --- un'estensione del supporto base di Nema, è implementato da questa MCU.

L'MCU è dotato di una periferica driver LCD che LVGL utilizza per aggiornare il display. Consultare la documentazione del driver [LTDC](#) per maggiori informazioni e per personalizzarne l'utilizzo.

#### Display per Single-Board Computer

I Single-Board Computer Displays sono pronti per l'uso con le schede carrier Dahlia e Mallow [Toradex](#). Infatti, queste schede carrier sono compatibili con tutti i display Riverdi MIPI-DSI a 34 pin.

## Altri prodotti

Gli **EVE Intelligent Display** sono dotati di un controller grafico IC Bridgetek EVE, in modo che il display possa essere controllato utilizzando comandi di disegno di alto livello su un'interfaccia a larghezza di banda inferiore rispetto a RGB, MIPI, ecc.

Gli **HDMI Displays** sono dotati di un'interfaccia HDMI per un'integrazione semplificata con PC o SBC. sport an HDMI interface for streamlined integration with PCs or SBCs. Questi display hanno comunque un profilo basso, quindi possono essere integrati facilmente in dispositivi personalizzati.

Le **Evaluation Boards** sono schede di supporto convenienti per far funzionare i pannelli di visualizzazione Riverdi prima di progettare o acquistare una scheda di supporto per un'applicazione di produzione.

### 4.10.3 Viewe

Viewe è specializzata in soluzioni di visualizzazione complete da oltre 10 anni, con una notevole capacità di ricerca e sviluppo software e hardware, produzione in fabbrica e assistenza commerciale. Viewe non offre solo **soluzioni complete per display e touch** ma anche **display intelligenti embedded** con produzioni standard e personalizzate per trasformare i vostri progetti in realtà.

#### Portfolio Prodotti:

- Display Intelligenti Embedded
  - IOT\_AIOT Smart Display
  - Uart Smart Display
  - HDMI Display\_Raspberry Pi Display
  - Display Arduino

#### Display TFT | OLED

- Sunlight Readable Display
- AMOLED Displays
- Transflective TFT Displays
- Standard TFT Displays
- Bar Display/Stretched TFT Display
- Square TFT Display
- Round TFT Display

#### IOT AIOT Smart Displays

Gli Smart Display Viewe IOT AIOT sono una serie di display che utilizzano microcontrollori ESP32 con interazioni multimodali integrate di GUI/voce/telecamera, edge computing ML/AI e connettività fluida. Interfacce UART/RS232/RS485/CAN e interfacce IO ridondanti sono incluse per facilitarne l'utilizzo prolungato. LVGL/Arduino/ESP-IDF sono particolarmente adatti a queste schede.

#### lv\_port\_viewe\_7\_espifd

lv\_port\_viewe\_7\_espifd è un repository di porte LVGL che supporta lo Smart Display Viewe ESP32 S3 da 7 pollici. Si basa su ESP-IDF e utilizza il microcontrollore ESP32-S3.

- [Viewe ESP32 S3 Smart Display](#)

### **lv\_port\_viewe\_knob\_15\_espifdf**

`lv_port_viewe_knob_15_espifdf` è un "port repository" LVGL che supporta il display touch Knob AMOLED Viewe. Si basa su ESP-IDF e utilizza il microcontrollore ESP32-S3.

- Viewe AMOLED Touch Knob Display

### **Display Intelligenti UART**

I display intelligenti UART Viewe sono una serie di display che utilizzano microcontrollori ESP32 e STM32 con interfaccia UART/RS232/RS485/CAN e dispongono di interfaccia IO per la connessione alle periferiche. LVGL è particolarmente adatto a queste schede.

### **lv\_port\_viewe\_7\_espifdf**

`lv_port_viewe_7_espifdf` è un repository di porte LVGL che supporta lo Smart Display Viewe ESP32 S3 da 7 pollici. Si basa su ESP-IDF e utilizza il microcontrollore ESP32-S3.

- Viewe ESP32 S3 Smart Display

# CAPITOLO 5

## Funzionalità Comuni dei Widget

I seguenti dettagli si applicano a tutti i tipi di widget.

### 5.1 Nozioni di base sui Widget

#### 5.1.1 Cos'è un Widget?

Un Widget è l'**elemento costitutivo di base** dell'interfaccia utente LVGL.

Esempi di Widget: *Base Widget (and Screen), Button, Label, Image, List, Chart e Text Area*.

*Tutti i Widget* per vedere tutti i tipi di Widget.

Tutti i Widget sono referenziati utilizzando un puntatore `lv_obj_t` come handle. Questo puntatore può essere utilizzato in seguito per leggere o modificare gli attributi del Widget.

#### 5.1.2 Attributi

##### Attributi di base

Tutti i tipi di Widget condividono alcuni attributi di base:

- Posizione
- Dimensione
- Genitore
- Stili
- Eventi che genera
- Flag come *Clickable*, *Scollable*, ecc.
- Ecc.

È possibile impostare/ottenere questi attributi con le funzioni `lv_obj_set_...` e `lv_obj_get_...`. Ad esempio:

```
/* Set basic Widget attributes */
lv_obj_set_size(btn1, 100, 50); /* Set a button's size */
lv_obj_set_pos(btn1, 20,30); /* Set a button's position */
```

Per dettagli completi su posizione, dimensione, coordinate e layout, *Posizioni, Dimensioni e Layout*.

## Attributi specifici dei Widget

Anche i tipi di Widget hanno attributi speciali. Ad esempio, uno slider ha

- Valori minimi e massimi
- Valore corrente

Per questi attributi speciali, ogni tipo di Widget può avere funzioni API univoche. Ad esempio, per uno slider:

```
/* Set slider specific attributes */
lv_slider_set_range(slider1, 0, 100);           /* Set the min. and max. values */
lv_slider_set_value(slider1, 40, LV_ANIM_ON);    /* Set the current value (position) */
```

'API dei widget è descritta nella loro [Documentazione](#) ma è anche possibile consultare i rispettivi file header (ad esempio *widgets/lv\_slider.h*)

## Genitori e figli

Il genitore di un Widget viene impostato al momento della creazione del widget: il genitore viene passato alla funzione di creazione.

Per ottenere il genitore corrente di un Widget, si usa `lv_obj_get_parent(widget)`.

È possibile spostare il widget su un nuovo genitore con `lv_obj_set_parent(widget, new_parent)`.

Per ottenere un figlio specifico di un genitore, utilizzare `lv_obj_get_child(parent, idx)`. Alcuni esempi per `idx`:

- `0` preleva il primo figlio
- `1` preleva il secondo figlio
- `-1` preleva il figlio creato per ultimo

È possibile scorrere i figli di un Widget genitore in questo modo:

```
uint32_t i;
for(i = 0; i < lv_obj_get_child_count(parent); i++) {
    lv_obj_t * child = lv_obj_get_child(parent, i);
    /* Do something with child. */
}
```

`lv_obj_get_index(widget)` restituisce l'indice del Widget nel suo genitore. Equivale al numero di figli più grandi nel genitore.

È possibile portare un Widget in primo piano o in secondo piano con `lv_obj_move_foreground(widget)` e `lv_obj_move_background(widget)`.

È possibile modificare l'indice di un Widget nel suo genitore utilizzando `lv_obj_move_to_index(widget, index)`.

Si può scambiare la posizione di due Widget con `lv_obj_swap(widget1, widget2)`.

Per ottenere lo Screen di un Widget (genitore di livello più alto) si usa `lv_obj_get_screen(widget)`.

### 5.1.3 Meccanismi di Funzionamento

#### Struttura genitore-figlio

Un Widget genitore può essere considerato il contenitore dei suoi figli. Ogni Widget ha esattamente un Widget genitore (eccetto le "Screen" [schermate]), ma un Widget genitore può avere un numero qualsiasi di figli. Non ci sono limitazioni per il tipo di genitore, ma ci sono Widget che in genere sono un genitore (ad esempio un pulsante) o un figlio (ad esempio una "label" [etichetta]).

#### Spostamento simultaneo

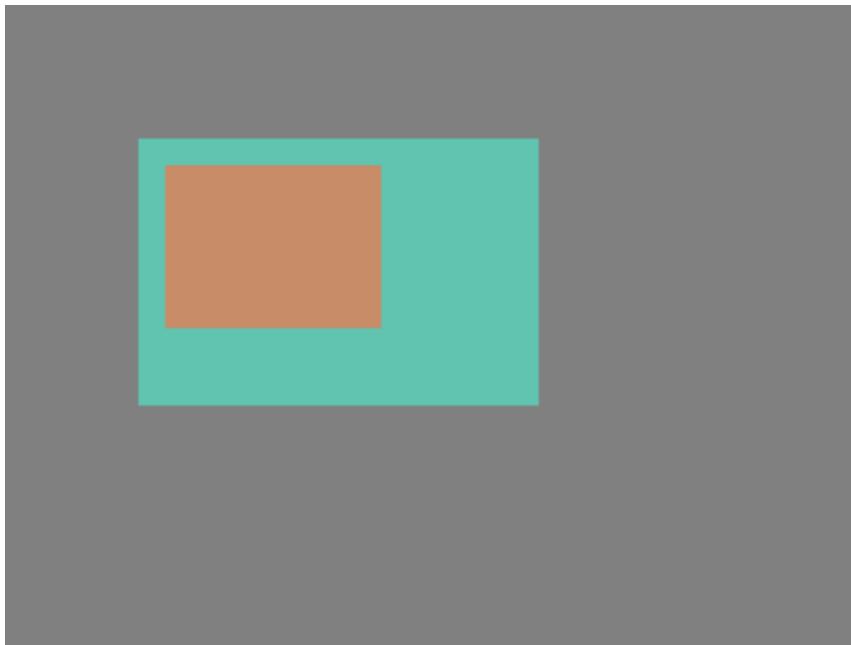
Se la posizione di un genitore cambia, i figli si sposteranno di conseguenza. Pertanto, tutte le posizioni sono relative al genitore.



```
lv_obj_t * parent = lv_obj_create(lv_screen_active()); /* Create a parent Widget on current Screen */
lv_obj_set_size(parent, 100, 80); /* Set size of parent */

lv_obj_t * widget1 = lv_obj_create(parent); /* Create a Widget on previously created parent Widget */
lv_obj_set_pos(widget1, 10, 10); /* Set position of new Widget */
```

Modificare la posizione del genitore:

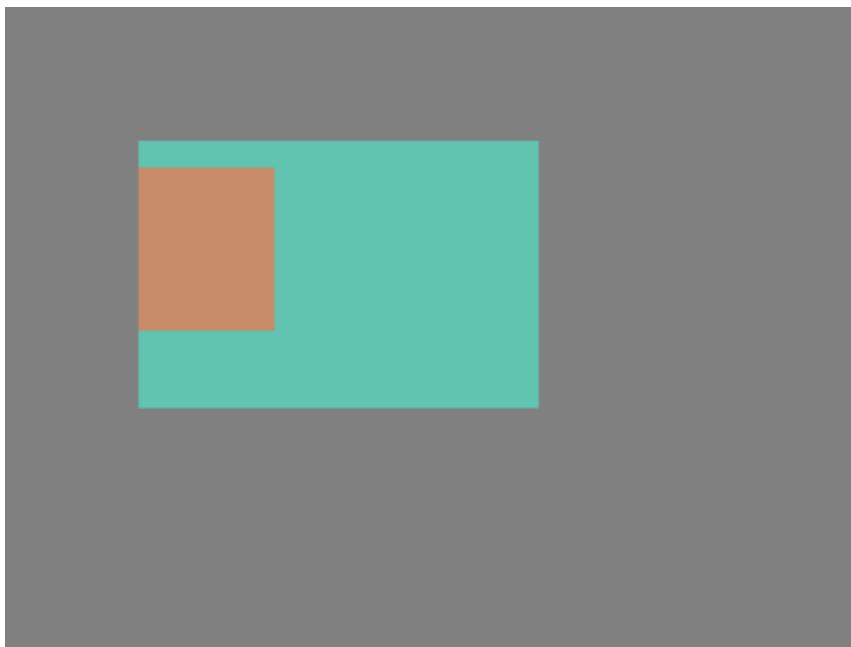


```
lv_obj_set_pos(parent, 50, 50); /* Move the parent. The child will move with it. */
```

(Per semplicità, la regolazione dei colori dei Widget non è mostrata nell'esempio).

## Visibilità solo sul genitore

Se un elemento figlio è parzialmente o completamente esterno al genitore, le parti esterne non saranno visibili.



```
lv_obj_set_x(widget1, -30); /* Move the child a little bit off the parent */
```

Questo comportamento può essere sovrascritto con `lv_obj_add_flag(widget, LV_OBJ_FLAG_OVERFLOW_VISIBLE)` che consente di disegnare gli elementi figlio all'esterno del genitore. Inoltre, è necessario registrare il seguente evento di callback (non richiesto nelle versioni precedenti).

Nota: `ext_width` deve essere la larghezza assoluta massima entro cui verranno disegnati gli elementi figlio.

```
static void ext_draw_size_event_cb(lv_event_t * e)
{
    lv_event_set_ext_draw_size(e, 30); /* Set 30px extra draw area around the widget */
}
```

## Creazione ed eliminazione di Widget

In LVGL, i Widget possono essere creati ed eliminati dinamicamente in fase di esecuzione. Ciò significa che solo i Widget attualmente creati (esistenti) consumano RAM.

Questo consente la creazione di uno Screen solo quando si clicca su un pulsante per aprirla e l'eliminazione degli Screen quando ne viene caricata una nuova.

Le interfacce utente possono essere create in base all'ambiente corrente del dispositivo. Ad esempio, è possibile creare misuratori, grafici, barre e cursori in base ai sensori attualmente collegati.

Ogni widget ha la sua funzione **create** con un prototipo come questo:

```
lv_obj_t * lv_<widget>_create(lv_obj_t * parent, <other parameters if any>);
```

In genere, le funzioni create hanno solo un parametro **parent** che indica su quale Widget creare il nuovo Widget.

Il valore restituito è un puntatore al Widget creato con tipo `lv_obj_t *`.

Esiste una funzione **delete** comune per tutti i tipi di Widget. Elimina il Widget e tutti i suoi elementi figlio.

```
void lv_obj_delete(lv_obj_t * widget);
```

`lv_obj_delete()` eliminerà immediatamente il Widget. Se per qualsiasi motivo non è possibile eliminare immediatamente il Widget, è possibile utilizzare `lv_obj_delete_async(widget)` che eseguirà l'eliminazione alla successiva chiamata di `lv_timer_handler()`. Questo è utile, ad esempio, se si desidera eliminare l'elemento genitore di un Widget nell'handler `LV_EVENT_DELETE`.

È possibile rimuovere tutti gli elementi figlio di un Widget (ma non il Widget stesso) utilizzando `lv_obj_clean(widget)`.

Si può usare `lv_obj_delete_delayed(widget, 1000)` per eliminare un widget dopo un certo periodo di tempo. Il ritardo è espresso in millisecondi.

A volte non si è sicuri che un widget sia stato eliminato e c'è bisogno di un modo per verificare se è ancora "alive". In qualsiasi momento prima che il Widget venga eliminato, si può usare `lv_obj_is_deleted(&widget)` per impostare il puntatore del Widget su `NULL` quando il Widget viene eliminato.

Assicurarsi che la variabile puntatore rimanga valida finché il Widget non viene eliminato. Ecco un esempio:

```
void some_timer_callback(lv_timer_t * t)
{
    static lv_obj_t * my_label;
    if(my_label == NULL) {
        my_label = lv_label_create(lv_screen_active());
        lv_obj_delete_delayed(my_label, 1000);
        lv_obj_null_on_delete(&my_label);
    } else {
        lv_obj_set_x(my_label, lv_obj_get_x(my_label) + 1);
    }
}
```

## 5.1.4 Screen

### Cosa sono gli Screen?

Da non confondere con un *Display* (`lv_display`), gli Screen sono semplicemente qualsiasi Widget creato senza un elemento genitore (ad esempio, passando `NULL` per l'argomento `parent` durante la creazione). In quanto tali, costituiscono la "root" [radice] di un "Widget Tree" [Albero dei Widget].

Normalmente, a questo scopo viene utilizzato il Widget Base, poiché dispone di tutte le funzionalità necessarie alla maggior parte degli Screen. Ma un Widget *Image* (`lv_image`) può anche essere utilizzato per creare uno sfondo per il "Widget Tree".

Tutti gli Screen:

- vengono automaticamente associati al *Display di Default* corrente al momento della creazione dello Screen;
- occupano automaticamente l'intera area del display associato;
- non possono essere spostati, ovvero funzioni come `lv_obj_set_pos()` e `lv_obj_set_size()` non possono essere utilizzate sugli Screen.

Ogni oggetto *Display* (`lv_display`) può avere più Screen [schermate] associate, ma non viceversa. Quindi la relazione:

```
Display
  |
  --- (one or more)
  / \
Screen Widgets (root of a Widget Tree)
  |
  0 (zero or more)
  / \
Child Widgets
```

### Creazione di Screen

Gli Screen vengono creati in questo modo:

```
lv_obj_t * scr1 = lv_obj_create(NULL);
```

Gli Screen possono essere eliminati con `lv_obj_delete(scr)`, ma assicurarsi di non eliminare *Lo Screen Attivo*.

### Lo Screen Attivo

Sebbene a ogni oggetto *Display* (`lv_display`) possa essere associato un numero qualsiasi di Widget Screen, solo uno di questi Screen è considerato "Attivo" in un dato momento. Quello Screen è definito lo "Screen Attivo" del Display. Per questo motivo, su un display verrà visualizzato solo uno Screen e i relativi Widget figli alla volta.

Quando ogni oggetto *Display* (`lv_display`) è stato creato, è stato creato uno Screen di default con esso e impostato come "Screen Attivo".

Per ottenere un puntatore allo "Screen Attivo", si chiama `lv_screen_active()`.

Per impostare uno Screen come "Screen Attivo", si chiama `lv_screen_load()` or `lv_screen_load_anim()`.

## Caricamento degli Screen

Per caricare un nuovo Screen, si usa `lv_screen_load(scr1)`. Questo imposta `scr1` come Screen Attivo.

## Caricare Screen con Opzioni Estese

Esiste un modo per caricare gli Screen che offre 2 opzioni aggiuntive (estese), consentendo al chiamante di specificare:

- un metodo di transizione opzionale e
- un'opzione per eliminare automaticamente la schermata visualizzata.

`lv_screen_load_anim(scr, transition_type, time, delay, auto_del)`. Esistono i seguenti tipi di transizione:

- `LV_SCREEN_LOAD_ANIM_NONE`: Cambia immediatamente dopo `delay` millisecondi
- `LV_SCREEN_LOAD_ANIM_OVER_LEFT`, `LV_SCREEN_LOAD_ANIM_OVER_RIGHT`, `LV_SCREEN_LOAD_ANIM_OVER_TOP` e `LV_SCREEN_LOAD_ANIM_OVER_BOTTOM`: Sposta il nuovo Screen sopra quello corrente verso la direzione indicata
- `LV_SCREEN_LOAD_ANIM_OUT_LEFT`, `LV_SCREEN_LOAD_ANIM_OUT_RIGHT`, `LV_SCREEN_LOAD_ANIM_OUT_TOP` e `LV_SCREEN_LOAD_ANIM_OUT_BOTTOM`: Sposta il vecchio Screen sopra quello corrente nella direzione indicata
- `LV_SCREEN_LOAD_ANIM_MOVE_LEFT`, `LV_SCREEN_LOAD_ANIM_MOVE_RIGHT`, `LV_SCREEN_LOAD_ANIM_MOVE_TOP` e `LV_SCREEN_LOAD_ANIM_MOVE_BOTTOM`: Sposta sia lo Screen corrente che quello nuovo nella direzione indicata
- `LV_SCREEN_LOAD_ANIM_FADE_IN` e `LV_SCREEN_LOAD_ANIM_FADE_OUT`: Dissolvenza del nuovo Screen su quello precedente, o viceversa.

Impostando `auto_del` a `true` il vecchio Screen verrà automaticamente eliminato al termine dell'animazione (se presente).

Il nuovo Screen diventerà attivo (restituito da `lv_screen_active()`) quando l'animazione inizia dopo il tempo `delay`. Tutti gli input sono disabilitati durante l'animazione dello Screen.

## I Layer

Quando viene creato un oggetto `lv_display_t`, vengono creati 4 Screen (layer) a cui vengono associati.

1. Layer Inferiore
2. Lo Screen Attivo
3. Layer Superiore
4. Layer di Sistema

1, 3 e 4 sono indipendenti da `Lo Screen Attivo` e verranno visualizzati (se contengono qualcosa di visibile) indipendentemente da quale sia lo Screen Attivo. `Screen Layers` e `Transparent Screens` per maggiori informazioni.

## 5.1.5 Parti

I widget sono costituiti da più parti. Ad esempio, un `Base Widget` utilizza le parti: "main" e la barra di scorrimento, mentre uno `Slider` utilizza le parti "main", l'indicatore e la manopola. Le parti sono simili agli *pseudo-elements* in CSS.

In LVGL sono presenti le seguenti parti predefinite:

- `LV_PART_MAIN`: Uno sfondo simile a un rettangolo

- *LV\_PART\_SCROLLBAR*: La/le barra/e di scorrimento
- *LV\_PART\_INDICATOR*: Indicatore, ad esempio per slider, barra, switch o la tick box della checkbox
- *LV\_PART\_KNOB*: Come una maniglia da afferrare per regolare il valore
- *LV\_PART\_SELECTED*: Indica l'opzione o la sezione attualmente selezionata
- *LV\_PART\_ITEMS*: Utilizzato se il widget ha più elementi simili (ad esempio, celle di tabella)
- *LV\_PART\_CURSOR*: Contrassegna un punto specifico, ad esempio il cursore dell'area di testo o del grafico
- *LV\_PART\_CUSTOM\_FIRST*: a qui è possibile aggiungere parti personalizzate.

Lo scopo principale delle parti è consentire di definire lo stile dei "componenti" dei widget. Sono descritte più dettagliatamente nella sezione *Panoramica sugli Stili*.

### 5.1.6 Stati

Il widget può trovarsi in una combinazione dei seguenti stati:

- *LV\_STATE\_DEFAULT*: Normale, stato rilasciato
- *LV\_STATE\_CHECKED*: Stato attivato o selezionato
- *LV\_STATE\_FOCUSED*: Focus tramite tastiera o encoder o clic tramite touchpad/mouse
- *LV\_STATE\_FOCUS\_KEY*: Focus tramite tastiera o encoder ma non tramite touchpad/mouse
- *LV\_STATE\_EDITED*: Modifica tramite encoder
- *LV\_STATE\_HOVERED*: Sovrapposto dal mouse (non supportato ora)
- *LV\_STATE\_PRESSED*: Premuto
- *LV\_STATE\_SCROLLED*: In corso di scrolling
- *LV\_STATE\_DISABLED*: Stato disabilitato
- *LV\_STATE\_USER\_1*: Stato personalizzato
- *LV\_STATE\_USER\_2*: Stato personalizzato
- *LV\_STATE\_USER\_3*: Stato personalizzato
- *LV\_STATE\_USER\_4*: Stato personalizzato

The states are usually automatically changed by the library as the user interacts with a Widget (presses, releases, focuses, etc.). However, the states can be changed manually as well. To set or clear given state (but leave the other states untouched) use `lv_obj_add_state(widget, LV_STATE_...)` and `lv_obj_remove_state(widget, LV_STATE_...)`. In both cases OR-ed state values can be used as well. E.g. `lv_obj_add_state(widget, LV_STATE_PRESSED | LV_PRESSED_CHECKED)`.

Per saperne di più sugli stati, leggere la sezione correlata della *Panoramica sugli Stili*.

### 5.1.7 I Flag

There are some Widget attributes which can be enabled/disabled by `lv_obj_add_flag(widget, LV_OBJ_FLAG_...)` and `lv_obj_remove_flag(widget, LV_OBJ_FLAG_...)`.

- *LV\_OBJ\_FLAG\_HIDDEN* Nasconde il Widget. (Come se non ci fosse affatto)
- *LV\_OBJ\_FLAG\_CLICKABLE* Rendi il Widget cliccabile dai dispositivi di input
- *LV\_OBJ\_FLAG\_CLICK\_FOCUSABLE* Aggiunge lo stato focused al Widget quando cliccato
- *LV\_OBJ\_FLAG\_CHECKABLE* Attiva/disattiva lo stato checked quando si clicca sul widget
- *LV\_OBJ\_FLAG\_SCROLLABLE* Rende il widget "scrollable"
- *LV\_OBJ\_FLAG\_SCROLL\_ELASTIC* Consente lo "scrolling" interno ma a velocità inferiore
- *LV\_OBJ\_FLAG\_SCROLL\_MOMENTUM* Fa sì che il Widget continui a "scrollare" quando viene "lanciato"

- *LV\_OBJ\_FLAG\_SCROLL\_ONE* Consente lo scrolling di un solo elemento figlio agganciabile
- *LV\_OBJ\_FLAG\_SCROLL\_CHAIN\_HOR* Consente la propagazione dello scorrimento orizzontale a un elemento genitore
- *LV\_OBJ\_FLAG\_SCROLL\_CHAIN\_VER* Consente la propagazione dello scorrimento verticale a un elemento genitore
- *LV\_OBJ\_FLAG\_SCROLL\_CHAIN* Semplice packaging per (*LV\_OBJ\_FLAG\_SCROLL\_CHAIN\_HOR* | *LV\_OBJ\_FLAG\_SCROLL\_CHAIN\_VER*)
- *LV\_OBJ\_FLAG\_SCROLL\_ON\_FOCUS* Scorre automaticamente il Widget per renderlo visibile quando ha il focus
- *LV\_OBJ\_FLAG\_SCROLL\_WITH\_ARROW* Consente lo scorrimento del Widget con i tasti freccia
- *LV\_OBJ\_FLAG\_SNAPPABLE* Se lo "scroll snap" è abilitato sul genitore, può agganciarsi a questo Widget
- *LV\_OBJ\_FLAG\_PRESS\_LOCK* Mantiene il Widget premuto anche se la pressione è stata spostata dal Widget
- *LV\_OBJ\_FLAG\_EVENT\_BUBBLE* Propaga gli eventi anche al genitore
- *LV\_OBJ\_FLAG\_EVENT\_TRICKLE* Propaga gli eventi anche ai figli
- *LV\_OBJ\_FLAG\_STATE\_TRICKLE* Propaga i cambi di stato anche ai figli
- *LV\_OBJ\_FLAG\_GESTURE\_BUBBLE* Propaga i gesti al genitore
- *LV\_OBJ\_FLAG\_ADV\_HITTEST* Consente di eseguire test di hit (clic) più accurati. Ad esempio tenendo conto degli angoli arrotondati
- *LV\_OBJ\_FLAG\_IGNORE\_LAYOUT* Imposta il Widget come non posizionato dai layout
- *LV\_OBJ\_FLAG\_FLOATING* Non "scrolle" il Widget quando il genitore "scrolle" e ignora il layout
- *LV\_OBJ\_FLAG\_SEND\_DRAW\_TASK\_EVENTS* Abilita l'invio di eventi *LV\_EVENT\_DRAW\_TASK\_ADDED*
- *LV\_OBJ\_FLAG\_OVERFLOW\_VISIBLE* Non ritagliare il contenuto del figlio al contorno del genitore
- *LV\_OBJ\_FLAG\_FLEX\_IN\_NEW\_TRACK* Avvia un nuovo "flex track" su questo elemento
- *LV\_OBJ\_FLAG\_LAYOUT\_1* Flag personalizzato, libero da usare da Layout
- *LV\_OBJ\_FLAG\_LAYOUT\_2* Flag personalizzato, libero da usare da Layout
- *LV\_OBJ\_FLAG\_WIDGET\_1* Flag personalizzato, utilizzabile liberamente dal widget
- *LV\_OBJ\_FLAG\_WIDGET\_2* Flag personalizzato, utilizzabile liberamente dal widget
- *LV\_OBJ\_FLAG\_USER\_1* Flag personalizzato, utilizzabile liberamente dall'utente
- *LV\_OBJ\_FLAG\_USER\_2* Flag personalizzato, utilizzabile liberamente dall'utente
- *LV\_OBJ\_FLAG\_USER\_3* Flag personalizzato, utilizzabile liberamente dall'utente
- *LV\_OBJ\_FLAG\_USER\_4* Flag personalizzato, utilizzabile liberamente dall'utente

Alcuni esempi:

```
/* Hide on Widget */
lv_obj_add_flag(widget, LV_OBJ_FLAG_HIDDEN);

/* Make a Widget non-clickable */
lv_obj_remove_flag(widget, LV_OBJ_FLAG_CLICKABLE);
```

## 5.1.8 Eventi del Widget di Base

### Eventi dai Dispositivi di input

- *LV\_EVENT\_PRESSED* Il Widget è stato premuto.
- *LV\_EVENT\_PRESSING* Il Widget è premuto (inviato continuamente durante la pressione).
- *LV\_EVENT\_PRESS\_LOST* Il Widget è ancora premuto, ma il cursore/dito è stato spostato fuori dal Widget.

- ***LV\_EVENT\_SHORT\_CLICKED*** Il Widget è stato premuto per un breve periodo di tempo, poi rilasciato. Non inviato se si "scrolla".
- ***LV\_EVENT\_SINGLE\_CLICKED*** Inviato per il primo clic breve entro una breve distanza e un breve intervallo di tempo.
- ***LV\_EVENT\_DOUBLE\_CLICKED*** Inviato per il secondo clic breve entro una breve distanza e un breve intervallo di tempo.
- ***LV\_EVENT\_TRIPLE\_CLICKED*** Inviato per il terzo clic breve entro una breve distanza e un breve intervallo di tempo.
- ***LV\_EVENT\_LONG\_PRESSED*** L'oggetto è stato premuto per almeno `long_press_time`. Non inviato se si "scrolla".
- ***LV\_EVENT\_LONG\_PRESSED\_REPEAT*** Inviato dopo `long_press_time` ogni `long_press_repeat_time` ms. Non inviato se si "scrolla".
- ***LV\_EVENT\_CLICKED*** Inviato al rilascio se non si scorre (indipendentemente dalla pressione prolungata).
- ***LV\_EVENT\_RELEASED*** Inviato ogni volta che il Widget viene rilasciato.
- ***LV\_EVENT\_SCROLL\_BEGIN*** Inizia lo "scrolling". Il parametro evento è un puntatore all'animazione dello "scroll". Può essere modificato.
- ***LV\_EVENT\_SCROLL\_THROW\_BEGIN*** Ricevuto all'inizio dello scorrimento.
- ***LV\_EVENT\_SCROLL\_END*** Scorrimento terminato.
- ***LV\_EVENT\_SCROLL*** Scrolling
- ***LV\_EVENT\_GESTURE*** È stato rilevato un gesto. Il gesto si ottiene con `lv_indev_get_gesture_dir(lv_indev_active())`;
- ***LV\_EVENT\_KEY*** Un tasto viene inviata al Widget. La chiave si ottiene con `lv_indev_get_key(lv_indev_active())`;
- ***LV\_EVENT\_FOCUSED*** Il widget ha ricevuto il focus,
- ***LV\_EVENT\_DFOCUSSED*** Il focus del widget è stato perso.
- ***LV\_EVENT\_LEAVE*** Il focus del widget è stato perso ma è ancora selezionato.
- ***LV\_EVENT\_HIT\_TEST*** Esegue un "hit-test" avanzato.

## Eventi Speciali

- ***LV\_EVENT\_VALUE\_CHANGED*** quando il flag `LV_OBJ_FLAG_CHECKABLE` è abilitato e il Widget è stato cliccato (durante la transizione da/verso lo stato selezionato)

## Eventi di Disegno

- ***LV\_EVENT\_DRAW\_MAIN*** Esecuzione del disegno della parte principale
- ***LV\_EVENT\_DRAW\_MAIN\_BEGIN*** Avvio del disegno della parte principale
- ***LV\_EVENT\_DRAW\_MAIN\_END*** Fine del disegno della parte principale
- ***LV\_EVENT\_DRAW\_POST*** Esecuzione della fase di post-disegno (quando tutti i figli sono stati disegnati)
- ***LV\_EVENT\_DRAW\_POST\_BEGIN*** Avvio della fase di post-disegno (quando tutti i figli sono stati disegnati)
- ***LV\_EVENT\_DRAW\_POST\_END*** Completamento della fase di post-disegno (quando tutti i figli sono stati disegnati)

## Altri Eventi

- *LV\_EVENT\_DELETE* L'oggetto è in fase di eliminazione
- *LV\_EVENT\_CHILD\_CHANGED* L'elemento figlio è stato rimosso, aggiunto o le sue dimensioni e posizione sono state modificate
- *LV\_EVENT\_CHILD\_CREATED* L'elemento figlio è stato creato, viene sempre inoltrato a tutti i genitori
- *LV\_EVENT\_CHILD\_DELETED* L'elemento figlio è stato eliminato, viene sempre inoltrato a tutti i genitori
- *LV\_EVENT\_SIZE\_CHANGED* Le coordinate/dimensioni dell'oggetto sono cambiate
- *LV\_EVENT\_STYLE\_CHANGED* Lo stile dell'oggetto è cambiato
- *LV\_EVENT\_LAYOUT\_CHANGED* La posizione di un elemento figlio è cambiata a causa di un ricalcolo del layout (quando il contenitore ha uno stile di layout flex o grid)
- *LV\_EVENT\_GET\_SELF\_SIZE* Ottiene la dimensione interna di un widget

### Ulteriori Informazioni

Scoprire di più sugli *Eventi*.

## 5.1.9 Tasti

Se *LV\_OBJ\_FLAG\_CHECKABLE* è abilitato, *LV\_KEY\_RIGHT* e *LV\_KEY\_UP* rendono il Widget checked, e *LV\_KEY\_LEFT* e *LV\_KEY\_DOWN* lo rendono unchecked.

Se *LV\_OBJ\_FLAG\_SCROLLABLE* è abilitato, ma il Widget non è modificabile (come dichiarato dalla classe widget), i tasti freccia (*LV\_KEY\_UP*, *LV\_KEY\_DOWN*, *LV\_KEY\_LEFT*, *LV\_KEY\_RIGHT*) "scrollano" il Widget. Se il widget può scorrere solo verticalmente, *LV\_KEY\_LEFT* e *LV\_KEY\_RIGHT* scorreranno invece verso l'alto/verso il basso, rendendolo compatibile con un dispositivo di input encoder. Vedere *Input devices overview* per maggiori informazioni sui comportamenti degli encoder e sulla modalità di modifica.

### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

## 5.1.10 Nomi

Quando si crea un widget, il suo riferimento può essere memorizzato in una variabile puntatore *lv\_obj\_t*\*. Per utilizzare questo widget in più punti del codice, la variabile può essere passata come parametro di funzione o trasformata in variabile globale. Tuttavia, questo approccio presenta alcuni svantaggi:

- L'utilizzo di variabili globali non è pulito e generalmente non è raccomandato.
- Non è scalabile. Passare riferimenti a 20 widget come parametri di funzione non è l'ideale.
- È difficile verificare se un widget esiste ancora o è stato eliminato.

### Impostazione dei nomi

Per risolvere questi problemi, LVGL introduce un potente sistema di denominazione dei widget abilitabile impostando *LV\_USE\_OBJ\_NAME* in *lv\_conf.h*.

È possibile assegnare un nome personalizzato utilizzando *lv\_obj\_set\_name*(*obj*, "name") o *lv\_obj\_set\_name\_static*(*obj*, "name"). La variante "static" significa che il nome passato deve rimanere valido finché il widget esiste, poiché viene memorizzato solo il puntatore. In caso contrario, LVGL allocherà memoria per memorizzare una copia del nome.

Se un nome termina con #, LVGL lo sostituirà automaticamente con un indice basato sul numero di elementi simili con lo stesso nome di base. Se non viene specificato alcun nome, il valore predefinito è <widget\_type>\_#.

Di seguito è riportato un esempio che mostra come vengono risolti i nomi assegnati manualmente e automaticamente:

- Contenitore principale `lv_obj` chiamato "cont": "cont"
  - Contenitore `lv_obj` chiamato "header": "header"
    - \* `lv_label` senza nome: "lv\_label\_0"
    - \* `lv_label` chiamato "title": "title"
    - \* `lv_label` senza nome: "lv\_label\_1" (È la terza etichetta, ma i widget con nomi personalizzati non vengono conteggiati)
  - Contenitore `lv_obj` chiamato "buttons":
    - \* `lv_button` senza nome: "lv\_button\_0"
    - \* `lv_button` chiamato "second\_button": "second\_button"
    - \* `lv_button` senza nome: "lv\_button\_1"
    - \* `lv_button` chiamato `lv_button_#`: "lv\_button\_2"
    - \* `lv_button` chiamato `mybtn_#`: "mybtn\_0"
    - \* `lv_button` senza nome: "lv\_button\_2"
    - \* `lv_button` chiamato `mybtn_#`: "mybtn\_1"
    - \* `lv_button` chiamato `mybtn_#`: "mybtn\_2"
    - \* `lv_button` chiamato `mybtn_#`: "mybtn\_3"

## Ricerca dei widget

I widget possono essere trovati per nome in due modi:

1. Ottenere un figlio diretto per nome\* utilizzando `lv_obj_get_child_by_name(parent, "child_name")`. Esempio: `lv_obj_get_child_by_name(header, "title")` Si può anche utilizzare un "path" per trovare i figli annidati: `lv_obj_get_child_by_name(cont, "buttons/mybtn_2")`
2. Trovare un descendente a qualsiasi livello\* utilizzando `lv_obj_find_by_name(parent, "child_name")`. Esempio: `lv_obj_find_by_name(cont, "mybtn_1")` Notare che "mybtn\_1" è un elemento figlio di `buttons`, non un elemento figlio diretto di `cont`. Questo è utile quando si desidera ignorare la gerarchia e cercare solo per nome.

Poiché entrambe le funzioni iniziano la ricerca da un elemento padre specifico, è possibile avere più sottoalberi di widget con nomi identici sotto elementi padre diversi.

Per esempio, se `my_listitem_create(parent)` crea un widget denominato "list\_item\_#" con elementi secondari come "icon", "title", "ok\_button" e "lv\_label\_0", e viene chiamato 10 volte, è possibile trovare un "ok\_button" specifico in questo modo:

```
lv_obj_t * item = lv_obj_find_by_name(lv_screen_active(), "list_item_5");
lv_obj_t * ok_btn = lv_obj_find_by_name(item, "ok_button");

// Or
lv_obj_t * ok_btn = lv_obj_get_child_by_name(some_list_container, "list_item_5(ok_button");
```

I nomi vengono risolti **quando vengono recuperati**, non quando vengono impostati. Ciò significa che gli indici riflettono sempre lo stato corrente dell'albero dei widget al momento in cui il nome viene utilizzato.

### 5.1.11 Snapshot

È possibile generare un'immagine snapshot per un Widget insieme ai suoi elementi secondari. Verificare i dettagli in [Snapshot](#).

## 5.1.12 Esempio

### Oggetti base con stili personalizzati

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

void lv_example_obj_1(void)
{
    lv_obj_t * obj1;
    obj1 = lv_obj_create(lv_screen_active());
    lv_obj_set_size(obj1, 100, 50);
    lv_obj_align(obj1, LV_ALIGN_CENTER, -60, -30);

    static lv_style_t style_shadow;
    lv_style_init(&style_shadow);
    lv_style_set_shadow_width(&style_shadow, 10);
    lv_style_set_shadow_spread(&style_shadow, 5);
    lv_style_set_shadow_color(&style_shadow, lv_palette_main(LV_PALETTE_BLUE));

    lv_obj_t * obj2;
    obj2 = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj2, &style_shadow, 0);
    lv_obj_align(obj2, LV_ALIGN_CENTER, 60, 30);
}
#endif
```

### Creare un oggetto trascinabile

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

static void drag_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    lv_indev_t * indev = lv_indev_active();
    if(indev == NULL) return;

    lv_point_t vect;
    lv_indev_get_vect(indev, &vect);

    int32_t x = lv_obj_get_x_aligned(obj) + vect.x;
    int32_t y = lv_obj_get_y_aligned(obj) + vect.y;
    lv_obj_set_pos(obj, x, y);
}

/**
 * Make an object draggable.
 */
void lv_example_obj_2(void)
{
    lv_obj_t * obj;
    obj = lv_obj_create(lv_screen_active());
    lv_obj_set_size(obj, 150, 100);
    lv_obj_add_event_cb(obj, drag_event_handler, LV_EVENT_PRESSING, NULL);

    lv_label_t * label = lv_label_create(obj);
    lv_label_set_text(label, "Drag me");
    lv_obj_center(label);
}
#endif
```

### Trasformare un oggetto usando una matrice 3x3

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_DRAW_TRANSFORM_USE_MATRIX

static void timer_cb(lv_timer_t * timer)
{
    lv_obj_t * obj = (lv_obj_t *) lv_timer_get_user_data(timer);

    static float value = 0.1f;
    lv_matrix_t matrix;
    lv_matrix_identity(&matrix);
    lv_matrix_scale(&matrix, value, 1);
    lv_matrix_rotate(&matrix, value * 360);
    lv_obj_set_transform(obj, &matrix);

    value += 0.01f;

    if(value > 2.0f) {
        lv_obj_reset_transform(obj);
        value = 0.1f;
    }
}

void lv_example_obj_3(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_center(obj);

    lv_timer_create(timer_cb, 20, obj);
}

#else

void lv_example_obj_3(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text_static(label, "LV_DRAW_TRANSFORM_USE_MATRIX is not enabled");
    lv_obj_center(label);
}
```

(continues on next page)

(continua dalla pagina precedente)

```
}
```

```
#endif /*LV_DRAW_TRANSFORM_USE_MATRIX*/
#endif /*LV_BUILD_EXAMPLES*/
```

## 5.1.13 API

*lv\_obj.h*  
*lv\_obj\_class.h*  
*lv\_obj\_class\_private.h*  
*lv\_obj\_draw.h*  
*lv\_obj\_draw\_private.h*  
*lv\_obj\_event.h*  
*lv\_obj\_pos.h*  
*lv\_obj\_private.h*  
*lv\_obj\_scroll.h*  
*lv\_obj\_scroll\_private.h*  
*lv\_obj\_style.h*  
*lv\_obj\_style\_gen.h*  
*lv\_obj\_style\_private.h*  
*lv\_obj\_tree.h*  
*lv\_refr.h*  
*lv\_refr\_private.h*  
*lv\_display.h*  
*lv\_flex.h*  
*lv\_grid.h*  
*lv\_api\_map\_v8.h*  
*lv\_api\_map\_v9\_1.h*  
*lv\_observer.h*  
*lv\_xml\_obj\_parser.h*  
*lv\_obj\_property\_names.h*

## 5.2 Posizioni, Dimensioni e Layout

### 5.2.1 Panoramica

Similmente a molte altre parti di LVGL, il concetto di impostazione delle coordinate è stato ispirato dai CSS. LVGL non ha affatto un'implementazione completa dei CSS, ma ne implementa un sottoinsieme comparabile (a volte con piccole modifiche).

In breve, questo significa:

- Le coordinate impostate esplicitamente sono memorizzate negli stili (posizione, dimensione, layout, ecc.)
- supporta min-width, max-width, min-height, max-height
- dispone di unità di misura pixel, percentuale e "content"

- la coordinata x=0; y=0 indica l'angolo in alto a sinistra dell'elemento genitore più il padding sinistro/superiore più la larghezza del bordo
- width/height indica la dimensione completa, l'"area del contenuto" è più piccola con padding e larghezza del bordo
- sono supportati un sottoinsieme di layout flexbox e grid

## 5.2.2 Unità di Lunghezza

Quando si passano "unità di lunghezza" (anche note come "unità di distanza" o "unità di dimensione") come argomenti a funzioni che modificano posizione, dimensione, ecc., per rendere più pratico il layout dell'interfaccia utente, è possibile scegliere tra diversi tipi di unità di misura.

### **pixel**

Specificare la dimensione in pixel: un valore intero <`LV_COORD_MAX`> indica sempre pixel. Es.  
`lv_obj_set_x(btn, 10).`

### **percentuale**

Specificare la dimensione come percentuale della dimensione del widget genitore o di se stesso, a seconda della proprietà. `lv_pct(value)` converte `value` in una percentuale. Es. `lv_obj_set_width(btn, lv_pct(50))`. Per evitare il sovraccarico della chiamata a `lv_pct()`, è possibile utilizzare anche la macro `LV_PCT(x)` per ottenere lo stesso risultato. Si noti che quando si utilizza questa funzionalità, il valore viene *memorizzato come percentuale* in modo che, se/quando la dimensione del contenitore genitore (o un altro fattore di posizionamento) cambia, questo valore di stile mantenga dinamicamente il suo significato.

### **"Content" contenuto**

Specifica la dimensione in funzione degli elementi figlio del widget. La macro `LV_SIZE_CONTENT`: passata come valore di dimensione ha un significato speciale: significa impostare la larghezza e/o l'altezza di un Widget abbastanza grande da includere tutti i suoi elementi figlio. Questo è simile a `auto` in CSS. Ee. `lv_obj_set_width(btn, LV_SIZE_CONTENT)`.

### **pollici**

Specifica la dimensione come 1/160 di pollice, come se fossero pixel su un display a 160 DPI, anche se un display potrebbe avere un DPI diverso. Usare `lv_dpx(n)` o `LV_DPX(n)` per farlo. Esempi:

n	DPI	Pixel Salcolati
40	320	80 pixel per ottenere 1/4 di pollice
40	160	40 pixel per ottenere 1/4 di pollice
40	130	33 pixel per ottenere 1/4 di pollice
80	130	66 pixel per ottenere 1/2 pollice

Vedere DPI in [Attributi](#).

## 5.2.3 Modello di Boxing

LVGL segue il modello border-box di CSS. Il "box" di un Widget è costituito dalle seguenti parti:

### **bounding box**

larghezza/altezza degli elementi.

### **border width**

larghezza del bordo.

### **padding**

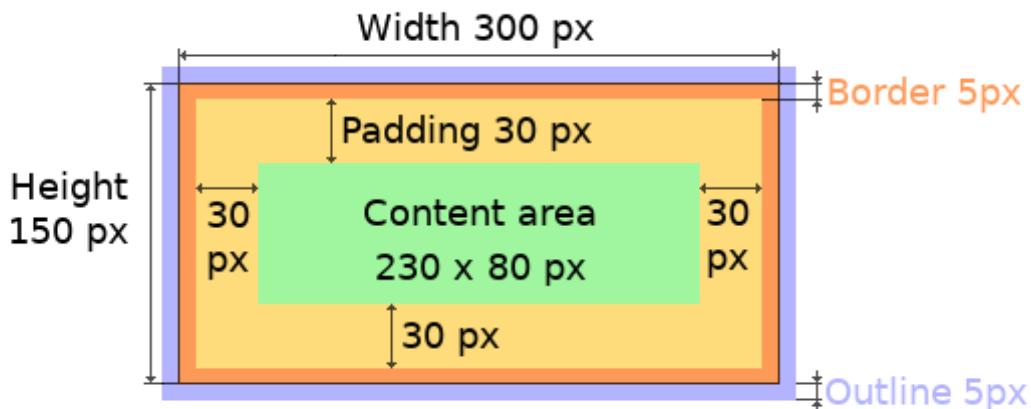
spazio tra i lati del Widget e i suoi elementi secondari.

### **margin**

spazio esterno al Widget (considerato solo da alcuni layout)

**content**

area del contenuto, ovvero la dimensione del bounding box ridotta della larghezza del bordo e del padding.



Il bordo viene disegnato all'interno del riquadro di delimitazione. All'interno del bordo, LVGL mantiene un "margin interno" quando posiziona i figli di un widget.

Il contorno viene disegnato all'esterno del riquadro di delimitazione.

#### 5.2.4 Note Importanti

Questa sezione descrive casi speciali in cui il comportamento di LVGL potrebbe essere inaspettato.

##### Calcolo delle coordinate posticipato

LVGL non ricalcola immediatamente tutte le modifiche alle coordinate. Questo viene fatto per migliorare le prestazioni. I Widget vengono invece contrassegnati come "dirty" [sporchi] e prima di ridisegnare lo schermo LVGL verifica se ci sono Widget "dirty". In tal caso, ne aggiorna posizione, dimensioni e layout.

In altre parole, se è necessario ottenere le coordinate di un widget e le coordinate sono appena state modificate, LVGL deve essere forzato a ricalcolare le coordinate. Per fare ciò, chiamare `lv_obj_update_layout()`.

Le dimensioni e la posizione potrebbero dipendere dal genitore o dal layout. Pertanto, `lv_obj_update_layout()` ricalcola le coordinate di tutti i Widget sullo schermo di `obj`.

##### Rimozione degli stili

Come descritto nella sezione [Utilizzo degli stili](#), le coordinate possono essere impostate anche tramite le proprietà di stile. Per essere più precisi, ogni proprietà relativa alle coordinate di stile viene memorizzata come proprietà di stile. Se si utilizza `lv_obj_set_x(widget, 20)` LVGL salva `x=20` nello stile locale del Widget.

Questo è un meccanismo interno e non ha molta importanza quando si utilizza LVGL. Tuttavia, c'è un caso in cui è necessario essere a conoscenza dell'implementazione. Se lo/gli stile/i di un Widget vengono rimossi da

```
lv_obj_remove_style_all(widget)
```

o

```
lv_obj_remove_style(widget, NULL, LV_PART_MAIN);
```

anche le coordinate impostate in precedenza verranno rimosse.

Ad esempio:

```
/* The size of obj1 will be set back to the default in the end */
lv_obj_set_size(widget1, 200, 100); /* Now obj1 has 200;100 size */
lv_obj_remove_style_all(widget1); /* It removes the set sizes */

/* widget2 will have 200;100 size in the end */
lv_obj_remove_style_all(widget2);
lv_obj_set_size(widget2, 200, 100);
```

## 5.2.5 Posizionamento dei Widget

### Diretto

Per impostare semplicemente le coordinate x e y di un Widget, utilizzare:

```
lv_obj_set_x(widget, 10);      /* Separate... */
lv_obj_set_y(widget, 20);
lv_obj_set_pos(widget, 10, 20); /* Or in one function */
```

Per default, le coordinate x e y vengono misurate dall'angolo in alto a sinistra dell'area del contenuto del genitore. Ad esempio, se il genitore ha cinque pixel di padding su ogni lato, il codice precedente posizionerà **obj** a (15, 25) perché l'area del contenuto inizia dopo il padding.

I valori percentuali vengono calcolati in base alle dimensioni dell'area del contenuto del genitore.

```
lv_obj_set_x(btn, lv_pct(10)); //x = 10 % of parent content area width
```

### Allineamento

#### All'interno del widget padre

In molti casi è più comodo dire a LVGL di allineare l'oggetto rispetto a un "anchor" [ancora] nel suo genitore *diversa* dal suo angolo superiore sinistro. Per stabilire tale "anchor", si chiama `lv_obj_set_align(widget, LV_ALIGN_...)`. Dopo questa chiamata, tale "ancora" verrà memorizzata finché non ne verrà stabilita un'altra. In altre parole, ogni futura impostazione x e y per quel Widget sarà relativa a tale "ancora".

Esempio: Posizionare il Widget (10,20) px rispetto al centro del suo genitore:

```
lv_obj_set_align(widget, LV_ALIGN_CENTER);
lv_obj_set_pos(widget, 10, 20);

/* Or combine the above in one function... */
lv_obj_align(widget, LV_ALIGN_CENTER, 10, 20);
```

È possibile utilizzare 9 "ancore" convenienti con queste funzioni:

- `LV_ALIGN_TOP_LEFT`
- `LV_ALIGN_TOP_MID`
- `LV_ALIGN_TOP_RIGHT`
- `LV_ALIGN_BOTTOM_LEFT`
- `LV_ALIGN_BOTTOM_MID`
- `LV_ALIGN_BOTTOM_RIGHT`
- `LV_ALIGN_LEFT_MID`
- `LV_ALIGN_RIGHT_MID`
- `LV_ALIGN_CENTER`

Vedere l'illustrazione qui sotto per visualizzare il significato.

È abbastanza comune allineare un elemento figlio al centro del suo elemento padre, pertanto esiste una funzione dedicata:

```
lv_obj_center(widget);
//Has the same effect
lv_obj_align(widget, LV_ALIGN_CENTER, 0, 0);
```

Se le dimensioni del padre cambiano, l'allineamento e la posizione impostati per gli elementi figlio vengono aggiornati automaticamente.

### Rispetto a un altro Widget

In alternativa, si può scegliere un'ancora su un altro Widget.

```
lv_obj_align_to(widget, reference_widget, align, x, y);
```

dove `align` può essere eseguito in uno dei seguenti modi:

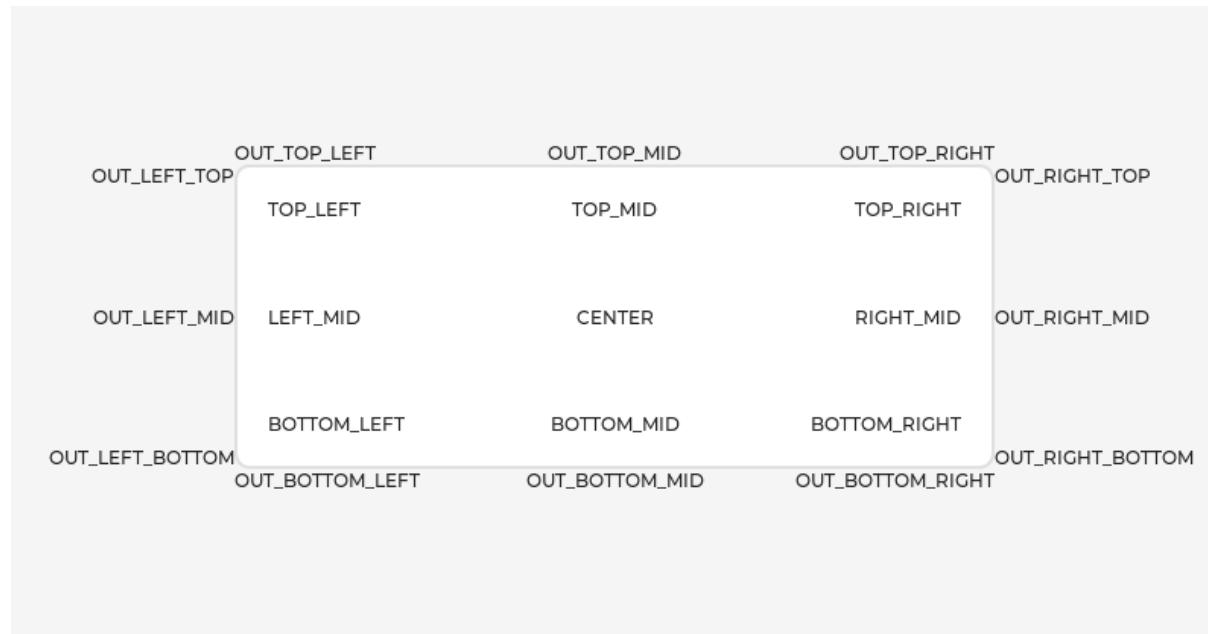
- `LV_ALIGN_OUT_TOP_LEFT`
- `LV_ALIGN_OUT_TOP_MID`
- `LV_ALIGN_OUT_TOP_RIGHT`
- `LV_ALIGN_OUT_BOTTOM_LEFT`
- `LV_ALIGN_OUT_BOTTOM_MID`
- `LV_ALIGN_OUT_BOTTOM_RIGHT`
- `LV_ALIGN_OUT_LEFT_TOP`
- `LV_ALIGN_OUT_LEFT_MID`
- `LV_ALIGN_OUT_LEFT_BOTTOM`
- `LV_ALIGN_OUT_RIGHT_TOP`
- `LV_ALIGN_OUT_RIGHT_MID`
- `LV_ALIGN_OUT_RIGHT_BOTTOM`

Esempio: per centrare orizzontalmente un'etichetta 10 pixel sopra un pulsante:

```
lv_obj_align_to(label, btn, LV_ALIGN_OUT_TOP_MID, 0, -10);
```

Si noti che, a differenza di `lv_obj_align()`, `lv_obj_align_to()` non ricorda l'"ancora" utilizzata e quindi non riposizionerà automaticamente il widget allineato se il widget di riferimento si sposta in seguito.

L'illustrazione seguente mostra il significato di ciascuna "ancora" menzionata sopra.



## 5.2.6 Dimensione

### Dimensionamento semplice

Anche la larghezza e l'altezza di un widget possono essere impostate facilmente:

```
lv_obj_set_width(widget, 200); /* Separate... */
lv_obj_set_height(widget, 100);
lv_obj_set_size(widget, 200, 100); /* Or in one function */
```

I valori percentuali vengono calcolati in base alle dimensioni dell'area del contenuto del genitore. Ad esempio, per impostare l'altezza del Widget all'altezza dello schermo:

```
lv_obj_set_height(widget, lv_pct(100));
```

Le impostazioni delle dimensioni supportano un valore speciale: *LV\_SIZE\_CONTENT*. Significa che le dimensioni del Widget nella rispettiva direzione saranno impostate in base alle dimensioni dei suoi elementi figlio. Si noti che verranno considerati solo gli elementi figlio sui lati destro e inferiore, mentre quelli in alto e a sinistra rimarranno ritagliati. Questa limitazione rende il comportamento più prevedibile.

I Widgets con *LV\_OBJ\_FLAG\_HIDDEN* o *LV\_OBJ\_FLAG\_FLOATING* verranno ignorati dal calcolo di *LV\_SIZE\_CONTENT*.

Le funzioni sopra indicate impostano la dimensione del riquadro di delimitazione di un widget, ma è possibile impostare anche la dimensione dell'area del contenuto. Ciò significa che il riquadro di delimitazione di un widget verrà ingrandito con l'aggiunta di un padding.

```
lv_obj_set_content_width(widget, 50); /* The actual width: padding left + 50 + padding right */
lv_obj_set_content_height(widget, 30); /* The actual width: padding top + 30 + padding bottom */
```

Le dimensioni del riquadro di delimitazione e dell'area del contenuto possono essere recuperate con le seguenti funzioni:

```
int32_t w = lv_obj_get_width(widget);
int32_t h = lv_obj_get_height(widget);
int32_t content_w = lv_obj_get_content_width(widget);
int32_t content_h = lv_obj_get_content_height(widget);
```

### Estensione dell'area di clic

Per default, i Widgets possono essere cliccati solo all'interno della loro area di delimitazione. Tuttavia, soprattutto con Widget di piccole dimensioni, può essere utile ingrandire l'area "cliccabile" di un Widget. Lo si può fare con *lv\_obj\_set\_ext\_click\_area(widget, size)*.

## 5.2.7 Utilizzo degli stili

In realtà, le proprietà di posizione, dimensione e allineamento sono proprietà di stile. Le "funzioni semplici" descritte sopra nascondono il codice relativo allo stile per semplicità e impostano le proprietà di posizione, dimensione e allineamento negli *local styles* del Widget.

Tuttavia, l'utilizzo degli stili per impostare le coordinate presenta alcuni grandi vantaggi:

- Semplifica l'impostazione di larghezza/altezza/etc. per più Widget contemporaneamente. Ad esempio, impostare tutti i cursori su una dimensione di 100x10 pixel.
- Rende inoltre possibile modificare i valori in un unico punto.
- I valori possono essere parzialmente sovrascritti da altri stili. Ad esempio, *style\_btn* impone il Widget su 100x50 per default ma aggiungendo *style\_full\_width* si sovrascrive solo la larghezza del Widget.
- Il Widget può avere posizione o dimensioni diverse a seconda dello stato. Ad esempio, 100 px di larghezza in *LV\_STATE\_DEFAULT* ma 120 px in *LV\_STATE\_PRESSED*.
- Le transizioni di stile possono essere utilizzate per rendere fluide le modifiche delle coordinate.

Ecco alcuni esempi per impostare le dimensioni di un widget utilizzando uno stile:

```
static lv_style_t style;
lv_style_init(&style);
lv_style_set_width(&style, 100);

lv_obj_t * btn = lv_button_create(lv_screen_active());
lv_obj_add_style(btn, &style, LV_PART_MAIN);
```

Come si vedrà di seguito, ci sono altre ottime funzionalità per l'impostazione di dimensioni e posizione. Tuttavia, per mantenere snella l'API LVGL, solo le funzionalità di impostazione delle coordinate più comuni hanno una versione "semplice", mentre le funzionalità più complesse possono essere utilizzate tramite stili.

### 5.2.8 Traslazione

Supponiamo che ci siano 3 pulsanti uno accanto all'altro. La loro posizione è impostata come descritto sopra. Ora si desidera spostare un pulsante leggermente verso l'alto quando viene premuto.

Un modo per ottenere questo risultato è impostare una nuova coordinata Y per lo stato premuto:

```
static lv_style_t style_normal;
lv_style_init(&style_normal);
lv_style_set_y(&style_normal, 100);

static lv_style_t style_pressed;
lv_style_init(&style_pressed);
lv_style_set_y(&style_pressed, 80);

lv_obj_add_style(btn1, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn1, &style_pressed, LV_STATE_PRESSED);

lv_obj_add_style(btn2, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn2, &style_pressed, LV_STATE_PRESSED);

lv_obj_add_style(btn3, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn3, &style_pressed, LV_STATE_PRESSED);
```

Funziona, ma non è molto flessibile perché la coordinata premuta è codificata. Se i pulsanti non sono a y=100, **style\_pressed** non funzionerà come previsto. Per risolvere questo problema, è possibile utilizzare le traslazioni:

```
static lv_style_t style_normal;
lv_style_init(&style_normal);
lv_style_set_y(&style_normal, 100);

static lv_style_t style_pressed;
lv_style_init(&style_pressed);
lv_style_set_translate_y(&style_pressed, -20);

lv_obj_add_style(btn1, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn1, &style_pressed, LV_STATE_PRESSED);

lv_obj_add_style(btn2, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn2, &style_pressed, LV_STATE_PRESSED);

lv_obj_add_style(btn3, &style_normal, LV_STATE_DEFAULT);
lv_obj_add_style(btn3, &style_pressed, LV_STATE_PRESSED);
```

La traslazione viene applicata dalla posizione corrente del Widget.

Anche i valori percentuali possono essere utilizzati nelle traslazioni. La percentuale è relativa alle dimensioni del Widget (e non alle dimensioni del widget principale). Ad esempio, **lv\_pct(50)** sposterà il Widget di metà della sua larghezza/altezza.

La traslazione viene applicata dopo il calcolo dei layout. Pertanto, anche la posizione dei Widget disposti può essere traslata.

La traslazione sposta effettivamente il widget. Ciò significa che fa sì che le barre di scorrimento e i Widget di dimensioni **LV\_SIZE\_CONTENT** reagiscano al cambio di posizione.

### 5.2.9 Trasformazione

Analogamente alla posizione, anche le dimensioni di un Widget possono essere modificate rispetto alle dimensioni correnti. La larghezza e l'altezza trasformate vengono aggiunte su entrambi i lati del Widget. Ciò significa che una larghezza trasformata di 10 px rende il Widget più largo di 2x10 pixel.

A differenza della traslazione della posizione, la trasformazione delle dimensioni non rende il Widget "realmente" più grande. In altre parole, barre di scorrimento, layout e **LV\_SIZE\_CONTENT** non reagiranno alle dimensioni trasformate. Quindi, la trasformazione delle dimensioni è "solo" un effetto visivo.

Questo codice ingrandisce un pulsante quando viene premuto:

```
static lv_style_t style_pressed;
lv_style_init(&style_pressed);
lv_style_set_transform_width(&style_pressed, 10);
lv_style_set_transform_height(&style_pressed, 10);
lv_obj_add_style(btn, &style_pressed, LV_STATE_PRESSED);
```

## Dimensioni Minime e Massime

Analogamente a CSS, LVGL supporta anche `min-width`, `max-width`, `min-height` e `max-height`. Si tratta di limiti che impediscono alle dimensioni di un Widget di diventare più piccole/più grandi di questi valori. Sono particolarmente utili se la dimensione è impostata in percentuale o [LV\\_SIZE\\_CONTENT](#).

```
static lv_style_t style_max_height;
lv_style_init(&style_max_height);
lv_style_set_max_height(&style_max_height, 200);

lv_obj_set_height(widget, lv_pct(100));
lv_obj_add_style(widget, &style_max_height, LV_STATE_DEFAULT); //Limit the height to 200 px
```

È possibile utilizzare anche valori percentuali relativi alle dimensioni dell'area del contenuto principale.

```
static lv_style_t style_max_height;
lv_style_init(&style_max_height);
lv_style_set_max_height(&style_max_height, lv_pct(50));

lv_obj_set_height(widget, lv_pct(100));
lv_obj_add_style(widget, &style_max_height, LV_STATE_DEFAULT); //Limit the height to half parent height
```

## 5.2.10 Layout

### Panoramica del layout

I layout possono aggiornare la posizione e le dimensioni degli elementi figlio di un widget. Possono essere utilizzati per disporre automaticamente gli elementi figlio in una riga o colonna, o in forme molto più complesse.

La posizione e le dimensioni impostate dal layout sovrascrivono le impostazioni "normali" di x, y, larghezza e altezza.

C'è una sola funzione che è la stessa per ogni layout: `lv_obj_set_layout()` (`widget, <LAYOUT_NAME>`) imposta il layout su un Widget. Per ulteriori impostazioni dell'elemento padre e degli elementi figlio, consultare la documentazione del layout in questione.

### Layout Integrati

LVGL include due layout molto potenti:

- Flexbox: organizza i widget in righe o colonne, con supporto per il wrapping e l'espansione degli elementi.
- Grid: organizza i widget in posizioni fisse in una tabella 2D.

Entrambi sono fortemente ispirati ai layout CSS con lo stesso nome. I layout sono descritti in dettaglio nella relativa sezione della documentazione.

### I Flag

Esistono alcuni flag che possono essere utilizzati sui Widget per influenzarne il comportamento con i layout:

- `LV_OBJ_FLAG_HIDDEN` I Widget nascosti vengono ignorati nei calcoli del layout.
- `LV_OBJ_FLAG_IGNORE_LAYOUT` Il Widget viene semplicemente ignorato dai layout. Le sue coordinate possono essere impostate come di consueto.
- `LV_OBJ_FLAG_FLOATING` Uguale a `LV_OBJ_FLAG_IGNORE_LAYOUT` ma il Widget con `LV_OBJ_FLAG_FLOATING` verrà ignorato nei calcoli di [LV\\_SIZE\\_CONTENT](#).

Questi flag possono essere aggiunti/rimossi con `lv_obj_add_flag(widget, FLAG)` e `lv_obj_remove_flag(widget, FLAG)`

## Aggiunta di nuovi layout

LVGL può essere liberamente esteso con un layout personalizzato come questo:

```
uint32_t MY_LAYOUT;
...
MY_LAYOUT = lv_layout_register(my_layout_update, &user_data);
...
void my_layout_update(lv_obj_t * widget, void * user_data)
{
    /* Will be called automatically if it's required to reposition/resize the children of "obj" */
}
```

È possibile aggiungere proprietà di stile personalizzate che possono essere recuperate e utilizzate nella callback di aggiornamento. Ad esempio:

```
uint32_t MY_PROP;
...
LV_STYLE_MY_PROP = lv_style_register_prop();
...
static inline void lv_style_set_my_prop(lv_style_t * style, uint32_t value)
{
    lv_style_value_t v = {
        .num = (int32_t)value
    };
    lv_style_set_prop(style, LV_STYLE_MY_PROP, v);
}
```

### 5.2.11 Esempi

### 5.2.12 API

## 5.3 I Layer

Quando il termine "layer" viene utilizzato nella documentazione LVGL, può riferirsi a diversi aspetti:

1. per i Widget, [Ordine di Creazione](#) crea una stratificazione naturale di Widget;
2. nel contesto del rendering pixel (disegno), ci sono [Draw Layers](#);
3. gli [Screen Layers](#) permanenti fanno parte di ciascun oggetto [Display \(lv\\_display\)](#).

Il punto 1 è trattato di seguito. I punti 2 e 3 sono trattati rispettivamente in [Draw Layers](#) e [Screen Layers](#).

### 5.3.1 Ordine di Creazione

Per default, LVGL disegna i nuovi Widget sopra i vecchi Widget.

Ad esempio, supponiamo di aggiungere un pulsante a un Widget padre denominato button1 e poi un altro pulsante denominato button2. Quindi button1 (insieme ai suoi Widget figlio/i) sarà sullo sfondo e potrà essere coperto da button2 e dai suoi figli.



```

/* Create a screen */
lv_obj_t * scr = lv_obj_create(NULL);
lv_screen_load(scr); /* Load the screen */

/* Create 2 buttons */
lv_obj_t * btn1 = lv_button_create(scr); /* Create the first button on the screen */
lv_obj_set_pos(btn1, 60, 40); /* Set the position of the first button */

lv_obj_t * btn2 = lv_button_create(scr); /* Create the second button on the screen */
lv_obj_set_pos(btn2, 180, 80); /* Set the position of the second button */

/* Add labels to the buttons */
lv_obj_t * label1 = lv_label_create(btn1); /* Create a label on the first button */
lv_label_set_text(label1, "Button 1"); /* Set the text of the label */

lv_obj_t * label2 = lv_label_create(btn2); /* Create a label on the second button */
lv_label_set_text(label2, "Button 2"); /* Set the text of the label */

/* Delete the second label */
lv_obj_delete(label2);

```

## Modifica dell'Ordine

Esistono quattro modi esplicativi per portare un Widget in primo piano:

- Si usa `lv_obj_move_foreground(widget)` per portare un Widget in primo piano. Allo stesso modo, si usa `lv_obj_move_background(widget)` per spostarlo in background.
- Si usa `lv_obj_move_to_index(widget, idx)` per spostare un widget a un indice specificato nell'ordine dei figli.
  - 0: background [sfondo]
  - `child_num - 1`: primo piano
  - `< 0`: conta dall'alto, per andare avanti (verso l'alto): `lv_obj_move_to_index(widget, lv_obj_get_index(widget) - 1)`
- Si usa `lv_obj_swap(widget1, widget2)` per scambiare la posizione relativa del livello di due widget.
- Quando si usa `lv_obj_set_parent(widget, new_parent)`, widget sarà in primo piano rispetto a `new_parent`.

### 5.3.2 API

`lv_obj_tree.h`

`lv_api_map_v8.h`

## 5.4 Stili

### 5.4.1 Panoramica sugli Stili

Gli stili vengono utilizzati per impostare l'aspetto dei Widget. Gli stili in LVGL sono fortemente ispirati ai CSS. Il concetto in breve è che uno stile è una variabile `lv_style_t` che può contenere proprietà come la larghezza del bordo, il font, il colore del testo e così via. È simile a una `class` in CSS.

- Gli stili possono essere assegnati ai Widget per modificarne l'aspetto. Al momento dell'assegnazione, è possibile specificare la parte di destinazione (`pseudo-element` in CSS) e lo stato di destinazione (`pseudo-class` in CSS). Ad esempio, è possibile aggiungere `style_blue` alla manopola di uno slider quando è premuto.
- Lo stesso stile può essere utilizzato da un numero qualsiasi di widget.
- Gli stili possono essere sovrapposti, il che significa che è possibile assegnare più stili a un widget e ogni stile può avere proprietà diverse. Pertanto, non tutte le proprietà devono essere specificate in uno stile. LVGL cercherà una proprietà finché non verrà definita da uno stile oppure utilizzerà un valore predefinito se non è specificato da nessuno degli stili. Ad esempio, `style_btn` può generare un pulsante grigio predefinito e `style_btn_red` può aggiungere solo `background-color=red` per sovrascrivere il colore di sfondo.
- Lo stile aggiunto più di recente ha la precedenza. Ciò significa che se una proprietà è specificata in due stili, verrà utilizzato lo stile più recente nel widget.
- Alcune proprietà (ad esempio, il colore del testo) possono essere ereditate da uno o più elementi padre se non sono specificate in un Widget.
- I Widget possono anche avere *local styles* con precedenza maggiore rispetto agli stili "normali".
- A differenza di CSS (dove le `pseudo-classes` descrivono stati diversi, ad esempio `:focus`), in LVGL una proprietà viene assegnata a un determinato stato.
- Le transizioni possono essere applicate quando il Widget cambia stato.

#### Stati

I Widget possono trovarsi nella seguente combinazione di stati:

- `LV_STATE_DEFAULT`: (0x0000) Stato normale, rilasciato
- `LV_STATE_CHECKED`: (0x0001) Stato attivato o selezionato
- `LV_STATE_FOCUSED`: (0x0002) Focus tramite tastiera o encoder o clic tramite touchpad/mouse
- `LV_STATE_FOCUS_KEY`: (0x0004) Focus tramite tastiera o encoder ma non tramite touchpad/mouse
- `LV_STATE_EDITED`: (0x0008) Modifica tramite encoder
- `LV_STATE_HOVERED`: (0x0010) Sovrapposto dal mouse
- `LV_STATE_PRESSED`: (0x0020) Premuto
- `LV_STATE_SCROLLED`: (0x0040) In scorrimento
- `LV_STATE_DISABLED`: (0x0080) Stato disabilitato
- `LV_STATE_USER_1`: (0x1000) Stato personalizzato
- `LV_STATE_USER_2`: (0x2000) Stato personalizzato
- `LV_STATE_USER_3`: (0x4000) Stato personalizzato
- `LV_STATE_USER_4`: (0x8000) Stato personalizzato

Un Widget può trovarsi in una combinazione di stati, ad esempio essere attivo e premuto contemporaneamente. Questo è rappresentato come `LV_STATE_FOCUSED` | `LV_STATE_PRESSED`.

A style can be added to any state or state combination. For example, setting a different background color for the default and pressed states. If a property is not defined in a state the best matching state's property will be used. Typically this means the property with `LV_STATE_DEFAULT` is used. If the property is not set even for the default state the default value will be used. (See later)

Poiché `LV_PART_MAIN` e `LV_STATE_DEFAULT` hanno entrambi valore zero, si può semplicemente passare `0` come argomento `selector` invece di `LV_PART_MAIN | LV_STATE_DEFAULT` come scorciatoia quando si aggiungono stili a un oggetto.

### Cosa significa "proprietà dello stato più corrispondente"

Gli stati hanno una precedenza, indicata dal loro valore (vedere l'elenco sopra). Un valore più alto significa una precedenza più alta. Per determinare quale proprietà dello stato utilizzare, prendiamo un esempio. Si immagini che il colore di sfondo sia definito in questo modo:

- `LV_STATE_DEFAULT`: bianco
- `LV_STATE_PRESSED`: grigio
- `LV_STATE_FOCUSED`: rosso

1. Inizialmente il Widget è nello stato di default, quindi è un caso semplice: la proprietà è perfettamente definita nello stato corrente del Widget, ovvero bianco.
2. Quando si preme il Widget, ci sono 2 proprietà correlate: il default con bianco (il default è relativo a tutti gli stati) e "pressed" con grigio. Lo stato "pressed" ha una precedenza `0x0020`, che è superiore alla precedenza `0x0000` dello stato di default, quindi verrà utilizzato il colore grigio.
3. Quando il widget è attivo, accade la stessa cosa che accade nello stato "pressed" e verrà utilizzato il colore rosso. (Lo stato "focused" ha una precedenza maggiore rispetto allo stato di default).
4. Quando il Widget ha il focus ed è premuto, funzionerebbero sia il grigio che il rosso, ma lo stato pressed ha una precedenza maggiore rispetto allo stato focused, quindi verrà utilizzato il colore grigio.
5. È possibile impostare, ad esempio, il colore rosa per `LV_STATE_PRESSED | LV_STATE_FOCUSED`. In questo caso, questo stato combinato ha una precedenza `0x0020 + 0x0002 = 0x0022`, che è superiore alla precedenza dello stato pressed, quindi verrà utilizzato il colore rosa.
6. Quando il Widget è nello stato checked, non esiste alcuna proprietà per impostare il colore di background per questo stato. Pertanto, in mancanza di un'opzione migliore, il Widget rimane bianco per la proprietà dello stato di default.

Alcune note pratiche:

- La precedenza (valore) degli stati è piuttosto intuitiva ed è qualcosa che l'utente si aspetterebbe naturalmente. Esempio: se un Widget ha il focus, l'utente vorrà comunque vedere se è premuto, quindi lo stato premuto ha una precedenza maggiore. Se lo stato con focus avesse una precedenza maggiore, sovrascriverebbe il colore di "pressed".
- Per impostare una proprietà per tutti gli stati (ad esempio, il colore di sfondo rosso), è sufficiente impostarla sullo stato di default. Se il widget non riesce a trovare una proprietà per il suo stato attuale, tornerà alla proprietà dello stato di default.
- Utilizzare gli stati con OR per descrivere le proprietà in casi complessi (ad esempio `pressed + checked + focused`).
- Potrebbe essere una buona idea utilizzare elementi di stile diversi per stati diversi. Ad esempio, trovare colori di sfondo per gli stati released, pressed, checked + pressed, focused, focused + pressed, focused + pressed + checked, ecc. è piuttosto difficile. In alternativa, ad esempio, utilizzare il colore di sfondo per gli stati pressed e checked e indicare lo stato focused con un colore del bordo diverso.

### Stili a Cascata

Non è necessario impostare tutte le proprietà in un unico stile. È possibile aggiungere più stili a un Widget e fare in modo che quest'ultimo modifichi o estenda l'aspetto. Ad esempio, creare uno stile generico per i pulsanti grigi e creare uno nuovo per i pulsanti rossi, in cui viene impostato solo il nuovo colore di sfondo.

Questo è molto simile a quello dei CSS, quando le classi utilizzate sono elencate come `<div class=".btn .btn-red">`.

Gli stili aggiunti in seguito hanno la precedenza su quelli impostati in precedenza. Quindi, nell'esempio del pulsante grigio/rosso sopra, lo stile del pulsante normale dovrebbe essere aggiunto per primo e quello rosso per secondo. Tuttavia, la precedenza degli stati viene comunque presa in considerazione. Esaminiamo il seguente caso:

- lo stile base del pulsante definisce il colore grigio scuro per lo stato di default e il colore grigio chiaro per lo stato pressed.
- lo stile del pulsante rosso definisce il colore di sfondo come rosso solo nello stato di default.

In questo caso, quando il pulsante viene rilasciato (è nello stato di default), sarà rosso perché è stata trovata una corrispondenza perfetta nello stile aggiunto più di recente (rosso). Quando il pulsante viene premuto, il colore grigio chiaro è una corrispondenza migliore perché descrive perfettamente lo stato corrente, quindi il pulsante sarà grigio chiaro.

## Ereditarietà

Alcune proprietà (in genere quelle relative al testo) possono essere ereditate dagli stili del Widget genitore. L'ereditarietà viene applicata solo se la proprietà specificata non è impostata negli stili del Widget (nemmeno nello stato di default). In questo caso, se la proprietà è ereditabile, il valore della proprietà verrà ricercato nella gerarchia padre finché un Widget non ne specifica un valore. I genitori utilizzeranno il proprio stato per determinare il valore. Quindi, un pulsante è "pressed" [premuto] e il colore del testo proviene da un genitore, verrà utilizzato il colore del testo "pressed".

## Parti

I Widget possono essere composti da *parti*, ciascuna delle quali può avere i propri stili.

In LVGL sono presenti le seguenti parti predefinite:

- *LV\_PART\_MAIN*: (0x000000) Uno background simile a un rettangolo
- *LV\_PART\_SCROLLBAR*: (0x010000) La/le barra/e di scorrimento
- *LV\_PART\_INDICATOR*: (0x020000) Indicatore, ad esempio per cursore, barra, interruttore o checkbox
- *LV\_PART\_KNOB*: (0x030000) Come una maniglia da afferrare per regolare un valore
- *LV\_PART\_SELECTED*: (0x040000) Indica l'opzione o la sezione attualmente selezionata
- *LV\_PART\_ITEMS*: (0x050000) Utilizzato se il widget ha più elementi simili (ad esempio, celle di tabella)
- *LV\_PART\_CURSOR*: (0x060000) Contrassegna un punto specifico, ad esempio il cursore dell'area di testo o del grafico
- *LV\_PART\_CUSTOM\_FIRST*: (0x080000) Da qui è possibile aggiungere identificatori di parti personalizzati.
- *LV\_PART\_ANY*: (0x0F0000) In alcune funzioni è possibile utilizzare un valore speciale per indirizzare tutte le parti.

Ad esempio, uno *Slider* ha tre parti:

- Main (background)
- Indicatore
- Knob [maniglia]

Ciò significa che tutte e tre le parti dello slider possono avere i propri stili. Più avanti si vedrà come aggiungere stili a Widget e a parti.

Poiché *LV\_PART\_MAIN* e *LV\_STATE\_DEFAULT* hanno entrambi valore zero, si può semplicemente passare 0 come argomento `selector` invece di *LV\_PART\_MAIN* | *LV\_STATE\_DEFAULT* come scorciatoia quando si aggiungono stili a un oggetto.

## Inizializzare gli Stili e Impostare/Ottenere Proprietà

Gli stili sono memorizzati nelle variabili `lv_style_t`. Le variabili di stile devono essere `static`, globali o allocate dinamicamente. In altre parole, non possono essere variabili locali nelle funzioni, che vengono eliminate all'uscita dalla funzione. Prima di utilizzare uno stile, è necessario inizializzarlo con `lv_style_init(&my_style)`. Dopo aver inizializzato uno stile, è possibile aggiungere o modificare proprietà.

Le funzioni di set di proprietà sono le seguenti: `lv_style_set_<property_name>(&style, <value>)`; Per esempio:

```
static lv_style_t style_btn;
lv_style_init(&style_btn);
lv_style_set_bg_color(&style_btn, lv_color_hex(0x115588));
lv_style_set_bg_opa(&style_btn, LV_OPA_50);
lv_style_set_border_width(&style_btn, 2);
lv_style_set_border_color(&style_btn, lv_color_black());

static lv_style_t style_btn_red;
lv_style_init(&style_btn_red);
lv_style_set_bg_color(&style_btn_red, lv_palette_main(LV_PALETTE_RED));
lv_style_set_bg_opa(&style_btn_red, LV_OPA_COVER);
```

Per rimuovere una proprietà, utilizzare:

```
lv_style_remove_prop(&style, LV_STYLE_BG_COLOR);
```

Per ottenere il valore di una proprietà da uno stile:

```
lv_style_value_t v;
lv_result_t res = lv_style_get_prop(&style, LV_STYLE_BG_COLOR, &v);
if(res == LV_RESULT_OK) { /* Found */
    do_something(v.color);
}
```

`lv_style_value_t` ha 3 campi, di cui solo uno verrà applicato, a seconda del tipo di proprietà a cui viene applicato:

- `num`: per le proprietà intere, booleane e di opacità
- `color`: per le proprietà relative al colore
- `ptr`: per le proprietà relative al puntatore

Per resettare uno stile (liberarne tutti i dati) utilizzare:

```
lv_style_reset(&style);
```

Gli stili possono essere creati anche come `const` per risparmiare RAM:

```
const lv_style_const_prop_t style1_props[] = {
    LV_STYLE_CONST_WIDTH(50),
    LV_STYLE_CONST_HEIGHT(50),
    LV_STYLE_CONST_PROPS_END
};

LV_STYLE_CONST_INIT(style1, style1_props);
```

In seguito, lo stile `const` potrà essere utilizzato come qualsiasi altro stile, ma (ovviamente) non sarà possibile aggiungere nuove proprietà.

## Aggiungere e rimuovere stili da un widget

Uno stile di per sé non ha effetto finché non viene aggiunto (assegnato) a un Widget.

### Aggiungere stili

Per aggiungere uno stile a un Widget, utilizzare `lv_obj_add_style(widget, &style, <selector>)`. `<selector>` è un valore OR tra le parti e lo stato a cui aggiungere lo stile. Alcuni esempi:

- `LV_PART_MAIN | LV_STATE_DEFAULT`
- `LV_STATE_PRESSED`: La parte "main" [principale] in stato premuto. `LV_PART_MAIN` può essere omesso
- `LV_PART_SCROLLBAR`: La parte della barra di scorrimento nello stato di default. `LV_STATE_DEFAULT` può essere omesso.

- `LV_PART_SCROLLBAR` | `LV_STATE_SCROLLED`: La parte della barra di scorrimento quando si scorre il widget.
- `LV_PART_INDICATOR` | `LV_STATE_PRESSED` | `LV_STATE_CHECKED` La parte dell'indicatore quando il widget viene premuto e selezionato contemporaneamente.

Uso di `lv_obj_add_style()`:

```
lv_obj_add_style(btn, &style_btn, 0);           /* Default button style */
lv_obj_add_style(btn, &btn_red, LV_STATE_PRESSED); /* Overwrite only some colors to red when pressed */
```

## Sostituzione di stili

Per sostituire uno stile specifico di un widget, si usa `lv_obj_replace_style(widget, old_style, new_style, selector)`. Questa funzione sostituirà `old_style` con `new_style` solo se `selector` corrisponde al `selector` usato in `lv_obj_add_style`. Sia `old_style` che `new_style` non devono essere `NULL`. Esistono funzioni separate per aggiungere e rimuovere stili. Se la combinazione di `old_style` e `selector` esiste più volte negli stili di `obj`, tutte le occorrenze verranno sostituite. Il valore restituito dalla funzione indica se è avvenuta almeno una sostituzione corretta.

Uso di `lv_obj_replace_style()`:

```
lv_obj_add_style(btn, &style_btn, 0);           /* Add a button style */
lv_obj_replace_style(btn, &style_btn, &new_style_btn, 0); /* Replace the button style with a different one */
```

## Rimozione di stili

Per rimuovere tutti gli stili da un widget, utilizzare `lv_obj_remove_style_all(widget)`.

Per rimuovere stili specifici, utilizzare `lv_obj_remove_style(widget, style, selector)`. Questa funzione rimuoverà `style` solo se il `selector` corrisponde al `selector` utilizzato in `lv_obj_add_style()`. `style` può essere `NULL` per controllare solo il `selector` e rimuovere tutti gli stili corrispondenti. Il `selector` può utilizzare i valori `LV_STATE_ANY` e `LV_PART_ANY` per rimuovere lo stile da qualsiasi stato o parte.

## Segnalazione delle modifiche di stile

Se uno stile già assegnato a un widget cambia (ad esempio, viene aggiunta o modificata una proprietà), i widget che utilizzano quello stile devono essere avvisati. Ci sono 3 opzioni per farlo:

1. Se si sa che le proprietà modificate possono essere applicate con un semplice ridisegno (ad esempio, modifiche di colore o opacità), si chiama semplicemente `lv_obj_invalidate(widget)` o `lv_obj_invalidate(lv_screen_active())`.
2. Se sono state modificate o aggiunte proprietà di stile più complesse e si sanno quali Widget sono interessati da quello stile, si chiama `lv_obj_refresh_style(widget, part, property)`. Per aggiornare tutte le parti e le proprietà, utilizzare `lv_obj_refresh_style(widget, LV_PART_ANY, LV_STYLE_PROP_ANY)`.
3. Per fare in modo che LVGL controlli tutti i Widget per vedere se utilizzano uno stile e aggiornarli quando necessario, chiamare `lv_obj_report_style_change(&style)`. Se `style` è `NULL` tutti i Widget riceveranno una notifica di modifica dello stile.

## Ottenere il valore di una proprietà di stile su un Widget

Per ottenere il valore finale di una proprietà di stile, considerando

- cascading,
- ereditarietà,
- stili locali e transizioni (vedere di seguito),

si possono utilizzare funzioni "get" di proprietà come questa: `lv_obj_get_style_<property_name>(widget, <part>)`. Queste funzioni utilizzano lo stato corrente del Widget e, se non esiste un candidato migliore, restituiscono il valore di default. Ad esempio:

```
lv_color_t color = lv_obj_get_style_bg_color(btn, LV_PART_MAIN);
```

## Stili Locali

Oltre agli stili "normali", i Widget possono anche memorizzare stili locali. Questo concetto è simile agli stili inline in CSS (ad esempio `<div style="color: red">`) con alcune modifiche.

Gli stili locali sono come gli stili normali, ma non possono essere condivisi tra altri Widget. Se utilizzati, gli stili locali vengono allocati automaticamente e liberati quando il Widget viene eliminato. Sono utili per aggiungere personalizzazioni locali a un widget.

A differenza di CSS, gli stili locali LVGL possono essere assegnati a stati (pseudo-classes) e parti (pseudo-elements).

Per impostare una proprietà locale, utilizzare funzioni come `lv_obj_set_style_<property_name>(widget, <value>, <selector>)`; Per esempio:

```
lv_obj_set_style_bg_color(slider, lv_color_red(), LV_PART_INDICATOR | LV_STATE_FOCUSED);
```

## Panoramica delle Proprietà di Stile

Per l'elenco completo delle proprietà di stile, cliccare [qui](#).

### Proprietà tipiche del background

Nella documentazione dei widget si troveranno frasi come "Il Widget \_\_\_\_\_ utilizza le tipiche proprietà di stile del background [sfondo]". Queste "proprietà tipiche del background" sono le proprietà a cui si fa riferimento:

- Background
- Border [bordatura]
- Outline [contorno]
- Shadow
- Padding [riempimento]
- Trasformazione di larghezza e altezza
- Traslazione X e Y

Consultare [Modello di Boxing](#) per il significato di questi termini.

## Transizioni

Per default, quando un Widget cambia stato (ad esempio, quando viene premuto), le nuove proprietà del nuovo stato vengono impostate immediatamente. Tuttavia, con le transizioni è possibile riprodurre un'animazione al cambio di stato. Ad esempio, premendo un pulsante, il suo colore di sfondo può essere animato per 300 ms fino a raggiungere il colore del pulsante premuto.

I parametri delle transizioni sono memorizzati negli stili. È possibile impostare

- il tempo della transizione
- il ritardo prima dell'inizio della transizione
- il path dell'animazione (noto anche come funzione di temporizzazione o easing)
- le proprietà da animare

Le proprietà della transizione possono essere definite per ogni stato. Ad esempio, impostando un tempo di transizione di 500 ms nello stato di default, quando il widget passa allo stato di default viene applicato un tempo di transizione di 500 ms. Impostando un tempo di transizione di 100 ms nello stato premuto, si ottiene una transizione di 100

ms quando si passa allo stato premuto. Questa configurazione di esempio comporta un rapido passaggio allo stato premuto e un lento ritorno allo stato di default.

Per descrivere una transizione, è necessario inizializzare una variabile `lv_transition_dsc_t` e aggiungerla a uno stile:

```
/* Only its pointer is saved so must static, global or dynamically allocated */
static const lv_style_prop_t trans_props[] = {
    LV_STYLE_BG_OPA, LV_STYLE_BG_COLOR,
    0, /* End marker */
};

static lv_style_transition_dsc_t transl;
lv_style_transition_dsc_init(&transl, trans_props, lv_anim_path_ease_out, duration_ms, delay_ms);
lv_style_set_transition(&style1, &transl);
```

## Opacità, Modalità di Blend [Fusione] e Trasformazioni

Se le proprietà `opa`, `blend_mode`, `transform_angle` o `transform_zoom` sono impostate su un valore non di default, LVGL crea un'istantanea del widget e dei suoi elementi secondari per fondere l'intero widget con le proprietà di opacità, modalità di fusione e trasformazione impostate.

Queste proprietà hanno questo effetto solo sulla parte **MAIN** del widget.

L'istantanea creata è chiamata "intermediate layer" [livello intermedio] o semplicemente "layer". Se solo `opa` e/o `blend_mode` sono impostate su un valore non predefinito, LVGL può creare il layer da blocchi più piccoli. La dimensione di questi blocchi può essere configurata tramite le seguenti proprietà in `lv_conf.h`:

- `LV_LAYER_SIMPLE_BUF_SIZE`: [bytes] la dimensione ottimale del buffer target. LVGL tenterà di allocare questa dimensione di memoria.
- `LV_LAYER_SIMPLE_FALLBACK_BUF_SIZE`: [bytes] utilizzati se `LV_LAYER_SIMPLE_BUF_SIZE` non è stato possibile allocare.

Se sono state utilizzate anche le proprietà di trasformazione, il layer non può essere renderizzato in blocchi, ma è necessario allocare un blocco di memoria più grande. La memoria richiesta dipende dai parametri di angolo, zoom e pivot e dalla dimensione dell'area da ridisegnare, ma non è mai maggiore della dimensione del widget (inclusa la dimensione di disegno aggiuntiva utilizzata per ombre, contorni, ecc.).

Se il widget può coprire completamente l'area da ridisegnare, LVGL crea un layer RGB (che è più veloce da rendere e utilizza meno memoria). Se è necessario utilizzare il rendering ARGB nel caso opposto, un widget potrebbe non coprire la sua area se ha raggio, `bg_opa < 255`, ha ombra, contorno, ecc.

Anche l'area di clic del widget viene trasformata di conseguenza.

## Filtro Colore

TODO [da fare]

## Temi

I temi sono una raccolta di stili. Se è presente un tema attivo, LVGL lo applica a ogni widget appena creato. Questo darà un aspetto predefinito all'interfaccia utente, che potrà poi essere modificato aggiungendo ulteriori stili.

Ogni display può avere un tema diverso. Ad esempio, è possibile avere un tema colorato su un TFT e un tema monocromatico su un display monocromatico secondario.

Per impostare un tema per un display, sono necessari due passaggi:

1. Inizializzare un tema
2. Assegnare il tema inizializzato a un display.

Le funzioni di inizializzazione del tema possono avere prototipi diversi. Questo esempio mostra come impostare il tema di "default":

```
lv_theme_t * th = lv_theme_default_init(display, /* Use DPI, size, etc. from this display */
                                         LV_COLOR_PALETTE_BLUE, /* Primary and secondary palette */
                                         LV_COLOR_PALETTE_CYAN,
                                         false, /* Dark theme? False = light theme. */
                                         &lv_font_montserrat_10, /* Small, normal, large fonts */
```

(continues on next page)

(continua dalla pagina precedente)

```
&lv_font_montserrat_14,
&lv_font_montserrat_18);

lv_display_set_theme(display, th); /* Assign theme to display */
```

I temi inclusi sono abilitati in `lv_conf.h`. Se il tema di default è abilitato da `LV_USE_THEME_DEFAULT` LVGL lo inizializza e lo imposta automaticamente quando viene creato un display.

## Estensione dei temi

I temi nativi possono essere estesi. Se viene creato un tema personalizzato, è possibile selezionare un tema genitore. Gli stili del tema genitore verranno aggiunti prima degli stili del tema personalizzato. È possibile concatenare un numero qualsiasi di temi in questo modo. Ad esempio, tema di default -> tema personalizzato -> tema scuro.

`lv_theme_set_parent(new_theme, base_theme)` estende `base_theme` con `new_theme`.

Di seguito è riportato un esempio.

## Esempi

### Stili Size

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/***
 * Using the Size, Position and Padding style properties
 */
void lv_example_style_1(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_width(&style, 150);
    lv_style_set_height(&style, LV_SIZE_CONTENT);

    lv_style_set_pad_ver(&style, 20);
    lv_style_set_pad_left(&style, 5);

    lv_style_set_x(&style, lv_pct(50));
    lv_style_set_y(&style, 80);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label, "Hello");
}

#endif
```

### Stili Background

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the background style properties
 */
void lv_example_style_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    static lv_grad_desc_t grad;
    grad.dir = LV_GRAD_DIR_VER;
    grad.stops_count = 2;
    grad.stops[0].color = lv_palette_lighten(LV_PALETTE_GREY, 1);
    grad.stops[0].opa = LV_OPA_COVER;
    grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
    grad.stops[1].opa = LV_OPA_COVER;

    /*Shift the gradient to the bottom*/
    grad.stops[0].frac = 128;
    grad.stops[1].frac = 192;

    lv_style_set_bg_grad(&style, &grad);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}
```

## Stili Border

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the border style properties
 */
void lv_example_style_3(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 10);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add border to the bottom+right*/
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style, 5);
    lv_style_set_border_opa(&style, LV_OPA_50);
    lv_style_set_border_side(&style, (lv_border_side_t)(LV_BORDER_SIDE_BOTTOM | LV_BORDER_SIDE_RIGHT));

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

## Stili Outline

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the outline style properties
 */
void lv_example_style_4(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add outline*/
    lv_style_set_outline_width(&style, 2);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_outline_pad(&style, 8);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

## Stili Shadow

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the Shadow style properties
 */
void lv_example_style_5(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add a shadow*/
    lv_style_set_shadow_width(&style, 55);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

## Stili Image

```
#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES && LV_USE_IMAGE

/**
 * Using the Image style properties
 */
void lv_example_style_6(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    lv_style_set_image_recolor(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_image_recolor_opa(&style, LV_OPA_50);
    lv_style_set_transform_rotation(&style, 300);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_image_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    LV_IMAGE_DECLARE(img_cogwheel_argb);
    lv_image_set_src(obj, &img_cogwheel_argb);

    lv_obj_center(obj);
}
#endif
```

## Stili Arc

```
#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES && LV_USE_ARC

/**
 * Using the Arc style properties
 */
void lv_example_style_7(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_arc_color(&style, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_arc_width(&style, 4);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_arc_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}
#endif
```

## Stili Text

```
#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES && LV_USE_LABEL

/**
 * Using the text style properties
 */
void lv_example_style_8(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 2));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_pad_all(&style, 10);

    lv_style_set_text_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_text_letter_space(&style, 5);
    lv_style_set_text_line_space(&style, 20);
    lv_style_set_text_decor(&style, LV_TEXT_DECOR_UNDERLINE);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_label_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_label_set_text(obj, "Text of\n"
                        "a label");

    lv_obj_center(obj);
}
#endif
```

## Stili Line

```
#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES || !LV_USE_LINE
    #error "This example requires LV_USE_LINE to be defined in lv_conf.h"
#endif

void lv_example_style_9(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_line_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_line_width(&style, 6);
    lv_style_set_line_rounded(&style, true);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_line_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    static lv_point_precise_t p[] = {{10, 30}, {30, 50}, {100, 0}};
    lv_line_set_points(obj, p, 3);

    lv_obj_center(obj);
}

#endif
```

## Transizione

```
#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES || !LV_USE_IMAGE
    #error "This example requires LV_USE_IMAGE to be defined in lv_conf.h"
#endif

void lv_example_style_10(void)
{
    static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, LV_STYLE_BORDER_COLOR, LV_STYLE_BORDER_WIDTH, 0};

    /* A default transition
     * Make it fast (100ms) and start with some delay (200 ms)*/
    static lv_style_transition_dsc_t trans_def;
    lv_style_transition_dsc_init(&trans_def, props, lv_anim_path_linear, 100, 200, NULL);

    /* A special transition when going to pressed state
     * Make it slow (500 ms) but start without delay*/
    static lv_style_transition_dsc_t trans_pr;
    lv_style_transition_dsc_init(&trans_pr, props, lv_anim_path_linear, 500, 0, NULL);

    static lv_style_t style_def;
    lv_style_init(&style_def);
    lv_style_set_transition(&style_def, &trans_def);

    static lv_style_t style_pr;
    lv_style_init(&style_pr);
    lv_style_set_bg_color(&style_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_border_width(&style_pr, 6);
    lv_style_set_border_color(&style_pr, lv_palette_darken(LV_PALETTE_RED, 3));
    lv_style_set_transition(&style_pr, &trans_pr);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style_def, 0);
    lv_obj_add_style(obj, &style_pr, LV_STATE_PRESSED);

    lv_obj_center(obj);
}
```

## Utilizzo di stili multipli

```
#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES || !LV_USE_IMAGE
    #error "This example requires LV_USE_IMAGE to be defined in lv_conf.h"
#endif

void lv_example_style_11(void)
{
    /*A base style*/
    static lv_style_t style_base;
    lv_style_init(&style_base);
    lv_style_set_bg_color(&style_base, lv_palette_main(LV_PALETTE_LIGHT_BLUE));
    lv_style_set_border_color(&style_base, lv_palette_darken(LV_PALETTE_LIGHT_BLUE, 3));
    lv_style_set_border_width(&style_base, 2);
    lv_style_set_radius(&style_base, 10);
    lv_style_set_shadow_width(&style_base, 10);
    lv_style_set_shadow_offset_y(&style_base, 5);
    lv_style_set_shadow_opa(&style_base, LV_OPA_50);
    lv_style_set_text_color(&style_base, lv_color_white());
    lv_style_set_width(&style_base, 100);
    lv_style_set_height(&style_base, LV_SIZE_CONTENT);

    /*Set only the properties that should be different*/
    static lv_style_t style_warning;
    lv_style_init(&style_warning);
    lv_style_set_bg_color(&style_warning, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_border_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW, 3));
    lv_style_set_text_color(&style_warning, lv_palette_darken(LV_PALETTE_YELLOW, 4));
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Create an object with the base style only*/
lv_obj_t * obj_base = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj_base, &style_base, 0);
lv_obj_align(obj_base, LV_ALIGN_LEFT_MID, 20, 0);

lv_obj_t * label = lv_label_create(obj_base);
lv_label_set_text(label, "Base");
lv_obj_center(label);

/*Create another object with the base style and earnings style too*/
lv_obj_t * obj_warning = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj_warning, &style_base, 0);
lv_obj_add_style(obj_warning, &style_warning, 0);
lv_obj_align(obj_warning, LV_ALIGN_RIGHT_MID, -20, 0);

label = lv_label_create(obj_warning);
lv_label_set_text(label, "Warning");
lv_obj_center(label);
}

#endif

```

## Stili Locali

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/** 
 * Local styles
 */
void lv_example_style_12(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style, lv_palette_lighten(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style, 3);

    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    /*Overwrite the background color locally*/
    lv_obj_set_style_bg_color(obj, lv_palette_main(LV_PALETTE_ORANGE), LV_PART_MAIN);
    lv_obj_center(obj);
}
#endif

```

## Aggiunta di stili a parti e a stati

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE

/** 
 * Add styles to parts and states
 */
void lv_example_style_13(void)
{
    static lv_style_t style_indic;
    lv_style_init(&style_indic);
    lv_style_set_bg_color(&style_indic, lv_palette_lighten(LV_PALETTE_RED, 3));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_HOR);

    static lv_style_t style_indic_pr;
    lv_style_init(&style_indic_pr);
    lv_style_set_shadow_color(&style_indic_pr, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_shadow_width(&style_indic_pr, 10);
    lv_style_set_shadow_spread(&style_indic_pr, 3);

    /*Create an object with the new style_pr*/
    lv_obj_t * obj = lv_slider_create(lv_screen_active());
    lv_obj_add_style(obj, &style_indic, LV_PART_INDICATOR);
    lv_obj_add_style(obj, &style_indic_pr, LV_PART_INDICATOR | LV_STATE_PRESSED);
    lv_slider_set_value(obj, 70, LV_ANIM_OFF);
    lv_obj_center(obj);
}
#endif

```

## Estensione del tema corrente

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_IMAGE
#include "../lvgl_private.h"

static lv_style_t style_btn;

/*Will be called when the styles of the base theme are already added
 to add new styles*/
static void new_theme_apply_cb(lv_theme_t * th, lv_obj_t * obj)
{
    LV_UNUSED(th);

    if(lv_obj_check_type(obj, &lv_button_class)) {
        lv_obj_add_style(obj, &style_btn, 0);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

}

static void new_theme_init_and_set(void)
{
    /*Initialize the styles*/
    lv_style_init(&style_btn);
    lv_style_set_bg_color(&style_btn, lv_palette_main(LV_PALETTE_GREEN));
    lv_style_set_border_color(&style_btn, lv_palette_darken(LV_PALETTE_GREEN, 3));
    lv_style_set_border_width(&style_btn, 3);

    /*Initialize the new theme from the current theme*/
    lv_theme_t * th_act = lv_display_get_theme(NULL);
    static lv_theme_t th_new;
    th_new = *th_act;

    /*Set the parent theme and the style apply callback for the new theme*/
    lv_theme_set_parent(&th_new, th_act);
    lv_theme_set_apply_cb(&th_new, new_theme_apply_cb);

    /*Assign the new theme to the current display*/
    lv_display_set_theme(NULL, &th_new);
}

/**
 * Extending the current theme
 */
void lv_example_style_14(void)
{
    lv_obj_t * btn;
    lv_obj_t * label;

    btn = lv_button_create(lv_screen_active());
    lv_obj_align(btn, LV_ALIGN_TOP_MID, 0, 20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Original theme");

    new_theme_init_and_set();

    btn = lv_button_create(lv_screen_active());
    lv_obj_align(btn, LV_ALIGN_BOTTOM_MID, 0, -20);

    label = lv_label_create(btn);
    lv_label_set_text(label, "New theme");
}

#endif

```

## Opacità e trasformazioni

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON && LV_USE_LABEL

/**
 * Opacity and Transformations
 */
void lv_example_style_15(void)
{
    lv_obj_t * btn;
    lv_obj_t * label;

    /*Normal button*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -70);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Normal");
    lv_obj_center(label);

    /*Set opacity
     *The button and the label is rendered to a layer first and that layer is blended*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_set_style_opa(btn, LV_OPA_50, 0);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 0);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Opa:50%");
    lv_obj_center(label);

    /*Set transformations
     *The button and the label is rendered to a layer first and that layer is transformed*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 40);
    lv_obj_set_style_rotation(btn, 150, 0);           /*15 deg*/
    lv_obj_set_style_transform_scale(btn, 256 + 64, 0); /*1.25x*/
    lv_obj_set_style_transform_pivot_x(btn, 50, 0);
    lv_obj_set_style_transform_pivot_y(btn, 20, 0);
    lv_obj_set_style_opa(btn, LV_OPA_50, 0);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 70);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Transf.");
    lv_obj_center(label);
}

#endif

```

## Manopola metallica con gradiente conico

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Simulate metallic knob using conical gradient
 * For best effect set LV_GRADIENT_MAX_STOPS to 8 or at least 3
 */
void lv_example_style_16(void)
{
#if LV_GRADIENT_MAX_STOPS >= 8
    static const lv_color_t grad_colors[8] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0xff, 0xff, 0xff),
        LV_COLOR_MAKE(0xfa, 0xfa, 0xfa),
        LV_COLOR_MAKE(0x79, 0x79, 0x79),
        LV_COLOR_MAKE(0x48, 0x48, 0x48),
        LV_COLOR_MAKE(0x4b, 0x4b, 0x4b),
        LV_COLOR_MAKE(0x70, 0x70, 0x70),
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
    };
#elif LV_GRADIENT_MAX_STOPS >= 3
    static const lv_color_t grad_colors[3] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0xff, 0xff, 0xff),
        LV_COLOR_MAKE(0x79, 0x79, 0x79),
    };
#else
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0x79, 0x79, 0x79),
    };
#endif

/*Create a style with gradient background and shadow*/
static lv_style_t style;
lv_style_init(&style);
lv_style_set_radius(&style, 500);
lv_style_set_bg_opa(&style, LV_OPA_COVER);
lv_style_set_shadow_color(&style, lv_color_black());
lv_style_set_shadow_width(&style, 50);
lv_style_set_shadow_offset_x(&style, 20);
lv_style_set_shadow_offset_y(&style, 20);
lv_style_set_shadow_opa(&style, LV_OPA_50);

/*First define a color gradient. In this example we use a gray color map with random values.*/
static lv_grad_dsc_t grad;

lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

/*Make a conical gradient with the center in the middle of the object*/
#if LV_GRADIENT_MAX_STOPS >= 8
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 0, 120, LV_GRAD_EXTEND_REFLECT);
#elif LV_GRADIENT_MAX_STOPS >= 3
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 125, LV_GRAD_EXTEND_REFLECT);
#else
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 110, LV_GRAD_EXTEND_REFLECT);
#endif

/*Set gradient as background*/
lv_style_set_bg_grad(&style, &grad);

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_set_size(obj, 200, 200);
lv_obj_center(obj);
}

#else

void lv_example_style_16(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/
```

## Gradiente radiale come sfondo

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Using radial gradient as background
 */
void lv_example_style_17(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x9B, 0x18, 0x42),
        LV_COLOR_MAKE(0x00, 0x00, 0x00),
    };

    int32_t width = lv_display_get_horizontal_resolution(NULL);
    int32_t height = lv_display_get_vertical_resolution(NULL);

    static lv_style_t style;
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_init(&style);

/*First define a color gradient. In this example we use a purple to black color map.*/
static lv_grad_dsc_t grad;

lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

/*Make a radial gradient with the center in the middle of the object, extending to the farthest corner*/
lv_grad_radial_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, LV_GRAD_RIGHT, LV_GRAD_BOTTOM, LV_GRAD_EXTEND_PAD);

/*Set gradient as background*/
lv_style_set_bg_grad(&style, &grad);

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_set_size(obj, width, height);
lv_obj_center(obj);
}

#else

void lv_example_style_17(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

## Gradienti per lo sfondo del pulsante

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Using various gradients for button background
 */
void lv_example_style_18(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x26, 0xa0, 0xda),
        LV_COLOR_MAKE(0x31, 0x47, 0x55),
    };

    /*Create a linear gradient going from the top left corner to the bottom at an angle, with reflected color map*/
    static lv_style_t style_with_linear_gradient_bg;
    static lv_grad_dsc_t linear_gradient_dsc; /*NOTE: the gradient descriptor must be static or global variable!*/
    lv_style_init(&style_with_linear_gradient_bg);
    lv_grad_init_stops(&linear_gradient_dsc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_linear_init(&linear_gradient_dsc, lv_pct(0), lv_pct(20), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_linear_gradient_bg, &linear_gradient_dsc);
    lv_style_set_bg_opa(&style_with_linear_gradient_bg, LV_OPA_COVER);

    /*Create a radial gradient with the center in the top left 1/3rd of the object, extending to the bottom right corner, with reflected color map*/
    static lv_style_t style_with_radial_gradient_bg;
    static lv_grad_dsc_t radial_gradient_dsc; /*NOTE: the gradient descriptor must be static or global variable!*/
    lv_style_init(&style_with_radial_gradient_bg);
    lv_grad_init_stops(&radial_gradient_dsc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_radial_init(&radial_gradient_dsc, lv_pct(30), lv_pct(30), lv_pct(100), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_radial_gradient_bg, &radial_gradient_dsc);
    lv_style_set_bg_opa(&style_with_radial_gradient_bg, LV_OPA_COVER);

    /*Create buttons with different gradient styles*/

    lv_obj_t * btn;
    lv_obj_t * label;

    /*Simple horizontal gradient/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
    lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
    lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_HOR, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -100);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Horizontal");
    lv_obj_center(label);

    /*Simple vertical gradient/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
    lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
    lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_VER, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -40);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Vertical");
    lv_obj_center(label);

    /*Complex linear gradient/
    btn = lv_button_create(lv_screen_active());
    lv_obj_add_style(btn, &style_with_linear_gradient_bg, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, 20);

    label = lv_label_create(btn);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_label_set_text(label, "Linear");
lv_obj_center(label);

/*Complex radial gradient*/
btn = lv_button_create(lv_screen_active());
lv_obj_add_style(btn, &style_with_radial_gradient_bg, 0);
lv_obj_set_size(btn, 150, 50);
lv_obj_align(btn, LV_ALIGN_CENTER, 0, 80);

label = lv_label_create(btn);
lv_label_set_text(label, "Radial");
lv_obj_center(label);
}

#else

void lv_example_style_18(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

## Test tra stile di ricolorazione o sfondo modale pieno

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SLIDER && LV_USE_LOG

/***
 * Test between a full background modal and a recolor modal
 */
void lv_example_style_19(void)
{
    /*Add lv_example_style_11 as background*/
    lv_example_style_11();

    /* Set to 1 to enable recolor overlay instead of solid background */
#if 0
    /* Apply a screen-wide tint using recolor (efficient overlay).
     * This modifies the visual appearance by blending a semi-transparent color
     * over existing content without creating additional objects.
     * It's lighter on performance compared to a full-size background object. */
    lv_obj_set_style_recolor(lv_screen_active(), lv_color_black(), 0);
    lv_obj_set_style_recolor_opa(lv_screen_active(), LV_OPA_50, 0);
#else
    /* Simulate a modal background by setting a semi-transparent black background
     * on lv_layer_top(), the highest built-in layer.
     * This method creates a new full-screen object and can consume more resources
     * compared to recolor, especially when using images or gradients. */
    lv_obj_set_style_bg_color(lv_layer_top(), lv_color_black(), 0);
    lv_obj_set_style_bg_opa(lv_layer_top(), LV_OPA_50, 0);
#endif

    lv_obj_t * obj = lv_slider_create(lv_layer_top());
    lv_obj_center(obj);

    lv_refr_now(NULL); /*Update layouts and render*/

    lv_obj_invalidate(lv_screen_active());

    uint32_t t = lv_tick_get();
    lv_refr_now(NULL); /*Render only*/
    LV_LOG_USER("%" LV_PRIu32 " ms\n", lv_tick_elaps(t));
}
#endif

```

## API

*lv\_obj.h*  
*lv\_obj\_pos.h*  
*lv\_obj\_scroll.h*  
*lv\_obj\_style.h*  
*lv\_obj\_style\_gen.h*  
*lv\_obj\_tree.h*  
*lv\_flex.h*  
*lv\_grid.h*  
*lv\_api\_map\_v8.h*  
*lv\_style.h*

*lv\_style\_gen.h*

## 5.4.2 Proprietà di Stile

### Dimensioni e posizione

Proprietà relative a dimensioni, posizione, allineamento e layout dei Widget.

#### width

Imposta la larghezza del widget. È possibile utilizzare i valori in pixel, percentuale e **LV\_SIZE\_CONTENT**. I valori percentuali sono relativi alla larghezza dell'area del "content" dell'elemento padre.

#### min\_width

Imposta una larghezza minima. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi alla larghezza dell'area del "content" dell'elemento padre.

#### max\_width

Imposta una larghezza massima. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi alla larghezza dell'area del "content" dell'elemento padre.

#### height

Imposta l'altezza del Widget. È possibile utilizzare i valori in pixel, percentuale e **LV\_SIZE\_CONTENT**. I valori percentuali sono relativi all'altezza dell'area del "content" dell'elemento padre.

#### min\_height

Imposta un'altezza minima. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi alla larghezza dell'area del "content" dell'elemento padre.

#### max\_height

Imposta un'altezza massima. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi all'altezza dell'area del "content" dell'elemento padre.

#### length

Il suo significato dipende dal tipo di Widget. Ad esempio, nel caso di `lv_scale`, indica la lunghezza dei tick.

#### x

Imposta la coordinata X del Widget considerando l'impostazione **align**. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi alla larghezza dell'area del "content" dell'elemento padre.

#### y

Imposta la coordinata Y del Widget considerando l'impostazione **align**. È possibile utilizzare i valori in pixel e in percentuale. I valori percentuali sono relativi all'altezza dell'area del "content" dell'elemento padre.

#### align

Imposta l'allineamento che indica da quale punto dell'oggetto padre devono essere interpretate le coordinate X e Y. I valori possibili sono: **LV\_ALIGN\_DEFAULT**, **LV\_ALIGN\_TOP\_LEFT/MID/RIGHT**, **LV\_ALIGN\_BOTTOM\_LEFT/MID/RIGHT**, **LV\_ALIGN\_LEFT/RIGHT\_MID**, **LV\_ALIGN\_CENTER**.

`LV_ALIGN_DEFAULT` significa `LV_ALIGN_TOP_LEFT` con direzione di base LTR e `LV_ALIGN_TOP_RIGHT` con direzione di base RTL.

#### **transform\_width**

Questo valore allarga il Widget su entrambi i lati. È possibile utilizzare valori in pixel e percentuali (con `lv_pct(x)`) Sposta il widget con questo valore nella direzione Y.

#### **transform\_height**

Questo valore alza il Widget su entrambi i lati. È possibile utilizzare valori in pixel e percentuali (con `lv_pct(x)`) I valori percentuali sono relativi all'altezza del Widget.

#### **translate\_x**

Sposta il Widget con questo valore nella direzione X. Applicato dopo layout, allineamenti e altri posizionamenti. È possibile utilizzare valori in pixel e percentuali (con `lv_pct(x)`) Sposta il widget con questo valore nella direzione Y.

#### **translate\_y**

Sposta il Widget con questo valore nella direzione Y. Applicato dopo layout, allineamenti e altri posizionamenti. È possibile utilizzare valori in pixel e percentuali (con `lv_pct(x)`) I valori percentuali sono relativi all'altezza del Widget.

#### **translate\_radial**

Sposta l'oggetto attorno al centro dell'oggetto padre (ad esempio attorno alla circonferenza di una scala)

#### **transform\_scale\_x**

Zoom orizzontale del Widget. Il valore 256 (o `LV_SCALE_NONE`) indica dimensioni normali, 128 dimensioni dimezzate, 512 dimensioni doppie e così via.

#### **transform\_scale\_y**

Zoom verticale del Widget. Il valore 256 (o `LV_SCALE_NONE`) indica dimensioni normali, 128 dimensioni dimezzate, 512 dimensioni doppie e così via.

#### **transform\_rotation**

Ruota il Widget. Il valore viene interpretato in unità di 0,1 gradi. Ad esempio, 450 significa 45 gradi.

#### **transform\_pivot\_x**

Imposta la coordinata X del punto pivot per le trasformazioni. Relativamente all'angolo in alto a sinistra del Widget.

#### **transform\_pivot\_y**

Imposta la coordinata Y del punto pivot per le trasformazioni. Relativamente all'angolo in alto a sinistra del Widget.

#### **transform\_skew\_x**

Inclina il Widget orizzontalmente. Il valore viene interpretato in unità di 0,1 gradi. Ad esempio, 450 significa 45 gradi.

### **transform\_skew\_y**

Inclina il Widget verticalmente. Il valore viene interpretato in unità di 0,1 gradi. Ad esempio, 450 significa 45 gradi.

### **Padding [riempimento]**

Proprietà che descrivono la spaziatura tra i lati del genitore e quelli dei figli e tra i figli. Molto simile alle proprietà di padding in HTML.

#### **pad\_top**

Imposta il padding in alto. Riduce l'area del contenuto in questa direzione.

#### **pad\_bottom**

Imposta il padding in basso. Riduce l'area del contenuto in questa direzione.

#### **pad\_left**

Imposta il padding a sinistra. Riduce l'area del contenuto in questa direzione.

#### **pad\_right**

Imposta il padding a destra. Riduce l'area del contenuto in questa direzione.

#### **pad\_row**

Imposta il padding tra le righe. Utilizzato dai layout.

#### **pad\_column**

Imposta il padding tra le colonne. Utilizzato dai layout.

#### **pad\_radial**

Allontana le etichette di testo dai segni di graduazione/dal resto di LV\_PART\_

### **Margin**

Proprietà per descrivere la spaziatura attorno a un Widget. Molto simile alle proprietà margin in HTML.

#### **margin\_top**

Imposta il margine in alto. Il widget manterrà questo spazio rispetto ai suoi "fratelli" nei layout.

#### **margin\_bottom**

Imposta il margine in basso. Il widget manterrà questo spazio rispetto ai suoi "fratelli" nei layout.

#### **margin\_left**

Imposta il margine a sinistra. Il widget manterrà questo spazio rispetto ai suoi "fratelli" nei layout.

#### **margin\_right**

Imposta il margine a destra. Il widget manterrà questo spazio rispetto ai suoi "fratelli" nei layout.

## Background

Proprietà per descrivere il colore di sfondo e l'immagine del Widget.

### bg\_color

Imposta il colore di sfondo del Widget.

### bg\_opa

Imposta l'opacità dello sfondo. Il valore 0, LV\_OPA\_0 o LV\_OPA\_TRANSP significa completamente trasparente, 255, LV\_OPA\_100 o LV\_OPA\_COVER significa completamente coprente, altri valori o LV\_OPA\_10, LV\_OPA\_20, ecc. significano semi-trasparenza.

### bg\_grad\_color

Imposta il colore del gradiente dello sfondo. Utilizzato solo se `grad_dir` non è LV\_GRAD\_DIR\_NONE

### bg\_grad\_dir

Imposta la direzione del gradiente dello sfondo. I valori possibili sono LV\_GRAD\_DIR\_NONE/HOR/VER.

### bg\_main\_stop

Imposta il punto da cui deve iniziare il colore di sfondo per i gradienti. 0 indica il lato superiore/sinistro, 255 il lato inferiore/destro, 128 il centro e così via.

### bg\_grad\_stop

Imposta il punto da cui deve iniziare il colore del gradiente dello sfondo. 0 indica il lato superiore/sinistro, 255 il lato inferiore/destro, 128 il centro e così via.

### bg\_main\_opa

Imposta l'opacità del primo colore del gradiente.

### bg\_grad\_opa

Imposta l'opacità del secondo colore del gradiente.

### bg\_grad

Imposta la definizione del gradiente. L'istanza puntata deve esistere mentre il Widget è attivo. NULL per disabilitare. Include BG\_GRAD\_COLOR, BG\_GRAD\_DIR, BG\_MAIN\_STOP e BG\_GRAD\_STOP in un unico descrittore e consente di creare gradienti anche con più colori. Se impostato, le altre proprietà relative al gradiente verranno ignorate.

### bg\_image\_src

Imposta un'immagine di sfondo. Può essere un puntatore a `lv_image_dsc_t`, un path a un file o un `LV_SYMBOL_...`

### bg\_image\_opa

Imposta l'opacità dell'immagine di sfondo. Il valore 0, LV\_OPA\_0 o LV\_OPA\_TRANSP significa completamente trasparente, 255, LV\_OPA\_100 o LV\_OPA\_COVER significa completamente coprente, altri valori o LV\_OPA\_10, LV\_OPA\_20, ecc. significano semi-trasparenza.

### **bg\_image\_recolor**

Imposta un colore da miscelare con l'immagine di sfondo.

### **bg\_image\_recolor\_opa**

Imposta l'intensità della ricolorazione dell'immagine di sfondo. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa nessuna miscelazione, 255, `LV_OPA_100` o `LV_OPA_COVER` significa ricolorazione completa, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. vengono interpretati proporzionalmente.

### **bg\_image\_tiled**

Se abilitato, l'immagine di sfondo verrà ricolorata. I valori possibili sono `true` o `false`.

## **Border [bordatura]**

Proprietà per descrivere i bordi

### **border\_color**

Imposta il colore del bordo

### **border\_opa**

Imposta l'opacità del bordo. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa completamente trasparente, 255, `LV_OPA_100` o `LV_OPA_COVER` significa completamente coprente, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. significano semi-trasparenza.

### **border\_width**

Imposta la larghezza del bordo. Possono essere utilizzati solo valori in pixel.

### **border\_side**

Imposta solo il lato o i lati su cui deve essere disegnato il bordo. I valori possibili sono `LV_BORDER_SIDE_NONE`/`TOP/BOTTOM/LEFT/RIGHT/INTERNAL`. È possibile utilizzare anche valori OR, ad esempio `LV_BORDER_SIDE_TOP | LV_BORDER_SIDE_LEFT`.

### **border\_post**

Imposta se il bordo deve essere disegnato prima o dopo i figli. `true`: dopo i figli, `false`: prima dei figli

## **Outline [contorno]**

Proprietà per descrivere il contorno. È come un bordo, ma disegnato all'esterno dei rettangoli.

### **outline\_width**

Imposta la larghezza del contorno in pixel.

### **outline\_color**

Imposta il colore del contorno.

**outline\_opa**

Imposta l'opacità del contorno. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa completamente trasparente, 255, `LV_OPA_100` o `LV_OPA_COVER` significa completamente coprente, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. significano semi-trasparenza.

**outline\_pad**

Imposta il riempimento del contorno, ovvero lo spazio tra il Widget e il contorno.

**Shadow**

Proprietà per descrivere l'ombra disegnata sotto i rettangoli.

**shadow\_width**

Imposta la larghezza dell'ombra in pixel. Il valore deve essere  $\geq 0$ .

**shadow\_offset\_x**

Imposta un offset dell'ombra in pixel nella direzione X.

**shadow\_offset\_y**

Imposta un offset dell'ombra in pixel nella direzione Y.

**shadow\_spread**

Esegue il calcolo dell'ombra per utilizzare un rettangolo più grande o più piccolo come base. Il valore può essere in pixel per ingrandire/ridurre l'area.

**shadow\_color**

Imposta il colore dell'ombra

**shadow\_opa**

Imposta l'opacità dell'ombra. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa completamente trasparente, 255, `LV_OPA_100` o `LV_OPA_COVER` significa completamente coprente, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. significano semi-trasparenza.

**Immagine**

Proprietà per descrivere le immagini

**image\_opa**

Imposta l'opacità di un'immagine. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa completamente trasparente, 255, `LV_OPA_100` o `LV_OPA_COVER` significa completamente coprente, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. significano semi-trasparenza.

**image\_recolor**

Imposta il colore da mescolare con l'immagine.

## image\_recolor\_opa

Imposta l'intensità della miscelazione dei colori. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa completamente trasparente, 255, `LV_OPA_100` o `LV_OPA_COVER` significa completamente coprente, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. significano semi-trasparenza.

## Line

Proprietà per descrivere i Widget lineari

### line\_width

Imposta la larghezza delle linee in pixel.

### line\_dash\_width

Imposta la larghezza dei trattini in pixel. Nota: il trattino funziona solo su linee orizzontali e verticali.

### line\_dash\_gap

Imposta la distanza tra i trattini in pixel. Nota: il trattino funziona solo su linee orizzontali e verticali.

### line\_rounded

Arrotonda le estremità delle linee. `true`: arrotondata, `false`: fine linea perpendicolare.

### line\_color

Imposta il colore delle linee.

### line\_opa

Imposta l'opacità delle linee.

## Arco

TODO [da fare]

### arc\_width

Imposta la larghezza (spessore) degli archi in pixel.

### arc\_rounded

Arrotonda le estremità degli archi. `true`: arrotondata, `false`: fine linea perpendicolare.

### arc\_color

Imposta il colore dell'arco.

### arc\_opa

Imposta l'opacità degli archi.

### **arc\_image\_src**

Imposta un'immagine da cui verrà mascherato l'arco. È utile per visualizzare effetti complessi sugli archi. Può essere un puntatore a `lv_image_dsc_t` o un percorso a un file.

## **Text**

Proprietà per descrivere le proprietà del testo. Tutte queste proprietà sono ereditate.

### **text\_color**

Imposta il colore del testo.

### **text\_opa**

Imposta l'opacità del testo. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa completamente trasparente, 255, `LV_OPA_100` o `LV_OPA_COVER` significa completamente coprente, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. significano semi-trasparenza.

### **text\_font**

Imposta il font del testo (un puntatore `lv_font_t *`).

### **text\_letter\_space**

Imposta la spaziatura delle lettere in pixel

### **text\_line\_space**

Imposta la spaziatura delle righe in pixel.

### **text\_decor**

Imposta la decorazione del testo. I valori possibili sono `LV_TEXT_DECOR_NONE/UNDERLINE/STRIKETHROUGH`. È possibile utilizzare anche valori con OR.

### **text\_align**

Imposta come allineare le righe del testo. Nota: non allinea il Widget stesso, ma solo le righe al suo interno. I valori possibili sono `LV_TEXT_ALIGN_LEFT/CENTER/RIGHT/AUTO`. `LV_TEXT_ALIGN_AUTO` rileva la direzione di base del testo e utilizza l'allineamento a sinistra o a destra di conseguenza.

### **text\_outline\_stroke\_color**

Imposta il colore del contorno delle lettere.

### **text\_outline\_stroke\_width**

Imposta la larghezza del tratto del contorno della lettera in pixel.

### **text\_outline\_stroke\_opa**

Imposta l'opacità del tratto del contorno della lettera. Il valore 0, `LV_OPA_0` o `LV_OPA_TRANSP` significa completamente trasparente, 255, `LV_OPA_100` o `LV_OPA_COVER` significa completamente coprente, altri valori o `LV_OPA_10`, `LV_OPA_20`, ecc. significano semi-trasparenza.

## Varie

Proprietà miste per vari scopi.

### radius

Imposta il raggio su ogni angolo. Il valore viene interpretato in pixel ( $\geq 0$ ) o LV\_RADIUS\_CIRCLE per il massimo. radius

### radial\_offset

Sposta il punto iniziale dell'oggetto (ad esempio, il segno di ridimensionamento) radialmente.

### clip\_corner

Abilita il ritaglio del contenuto in eccesso sull'angolo arrotondato. Può essere true o false.

### opa

Riduce tutti i valori di opacità del Widget di questo fattore. Il valore 0, LV\_OPA\_0 o LV\_OPA\_TRANSP significa completamente trasparente, 255, LV\_OPA\_100 o LV\_OPA\_COVER significa completamente coprente, altri valori o LV\_OPA\_10, LV\_OPA\_20, ecc. significano semi-trasparenza.

### opa\_layered

Per prima cosa disegna il Widget sul layer, poi riduce il fattore di opacità del layer. Il valore 0, LV\_OPA\_0 o LV\_OPA\_TRANSP significa completamente trasparente, 255, LV\_OPA\_100 o LV\_OPA\_COVER significa completamente coprente, altri valori o LV\_OPA\_10, LV\_OPA\_20, ecc. significano semi-trasparenza.

### color\_filter\_dsc

Mescola un colore con tutti i colori del Widget.

### color\_filter\_opa

Intensità di miscelazione del filtro colore.

### anim

Template di animazione per l'animazione del Widget. Dovrebbe essere un puntatore a lv\_anim\_t. I parametri di animazione sono specifici del widget, ad esempio il tempo di animazione potrebbe essere, ad esempio, il tempo di lampeggio del cursore sull'area di testo o il tempo di scorrimento di un roller. Consultare la documentazione dei Widget per saperne di più.

### anim\_duration

Durata dell'animazione in millisecondi. Il suo significato è specifico del widget. Ad esempio, il tempo di lampeggio del cursore sulla Text Area o il tempo di scorrimento di un roller. Consultare la documentazione dei Widget per saperne di più.

### transition

Un lv\_style\_transition\_desc\_t inizializzato per descrivere una transizione.

**blend\_mode**

Describe come sfumare i colori sullo sfondo. I valori possibili sono LV\_BLEND\_MODE\_NORMAL/ADDITIVE/SUBTRACTIVE/MULTIPLY/DIFFERENCE

**layout**

Imposta il layout del Widget. I figli verranno riposizionati e ridimensionati in base alle policy impostate per il layout. Per i valori possibili, consultare la documentazione dei layout.

**base\_dir**

Imposta la direzione di base del Widget. I valori possibili sono LV\_BIDI\_DIR\_LTR/RTL/AUTO.

**bitmap\_mask\_src**

Se impostato, verrà creato un layer per il widget e il layer verrà mascherato con questa maschera bitmap A8.

**rotary\_sensitivity**

Regola la sensibilità per i codificatori rotativi in unità 1/256. Significa: 128: rallenta la rotazione della metà, 512: raddoppia la velocità, 256: nessuna modifica

**Flex**

Proprietà del layout Flex.

**flex\_flow**

Definisce in quale direzione il layout Flex deve disporre i figli.

**flex\_main\_place**

Definisce come allineare i figli nella direzione del flusso del Flex.

**flex\_cross\_place**

Definisce come allineare i figli perpendicolarmente alla direzione del flusso del Flex.

**flex\_track\_place**

Definisce come allineare le tracce del flusso.

**flex\_grow**

Definisce quanto spazio occupare proporzionalmente dallo spazio libero della traccia del Widget.

**Grid**

Proprietà del layout della griglia.

**grid\_column\_dsc\_array**

Un array per descrivere le colonne della griglia. Dovrebbe essere terminato con LV\_GRID\_TEMPLATE\_LAST

### grid\_column\_align

Definisce come distribuire le colonne.

### grid\_row\_dsc\_array

Un array per descrivere le righe della griglia. Dovrebbe essere terminato con LV\_GRID\_TEMPLATE\_LAST

### grid\_row\_align

Definisce come distribuire le righe.

### grid\_cell\_column\_pos

Imposta la colonna in cui posizionare il Widget.

### grid\_cell\_x\_align

Imposta come allineare il Widget orizzontalmente.

### grid\_cell\_column\_span

Imposta il numero di colonne che il Widget deve occupare. Deve essere  $\geq 1$ .

### grid\_cell\_row\_pos

Imposta la riga in cui posizionare il widget.

### grid\_cell\_y\_align

Imposta come allineare il Widget verticalmente.

### grid\_cell\_row\_span

Imposta il numero di righe che il Widget deve occupare. Deve essere  $\geq 1$ .

## 5.5 Eventi

Gli eventi vengono attivati in LVGL quando accade qualcosa che potrebbe essere interessante per l'utente, ad esempio quando un Widget:

- viene cliccato
- viene fatto scorrere
- viene modificato il suo valore
- viene ridisegnato, ecc.

Oltre ai Widget, gli eventi possono essere registrati anche da display e dispositivi di input. Non viene spiegato in dettaglio di seguito, ma è possibile farlo modificando il prefisso delle funzioni da `lv_obj_` a `lv_display_` o `lv_indev_`.

### 5.5.1 Aggiunta di Eventi a un Widget

L'utente può assegnare funzioni di callback a un widget per elaborare gli eventi. In pratica, si presenta così:

```
lv_obj_t * btn = lv_button_create(lv_screen_active());
lv_obj_add_event_cb(btn, my_event_cb, LV_EVENT_CLICKED, user_data); /* Assign an event callback */

...
static void my_event_cb(lv_event_t * event)
{
    printf("Clicked\n");
}
```

Nell'esempio `LV_EVENT_CLICKED` significa che solo l'evento clic chiamerà `my_event_cb`. Vedere la [lista dei codici evento](#) per tutte le opzioni. `LV_EVENT_ALL` può essere utilizzato per ricevere tutti gli eventi.

L'ultimo parametro di `lv_obj_add_event_cb()` è un puntatore a eventuali dati personalizzati che saranno disponibili nell'evento. È possibile passare NULL per questo argomento se non è necessario utilizzare tali dati durante l'elaborazione dell'evento. È possibile recuperare il puntatore passato impostando la funzione di callback in questo modo:

```
my_user_data_t * user_data;
...
user_data = lv_event_get_user_data(e);
```

È possibile aggiungere altri eventi a un Widget, in questo modo:

```
lv_obj_add_event_cb(widget, my_event_cb_1, LV_EVENT_CLICKED, NULL);
lv_obj_add_event_cb(widget, my_event_cb_2, LV_EVENT_PRESSED, NULL);
lv_obj_add_event_cb(widget, my_event_cb_3, LV_EVENT_ALL); /* No filtering, receive all events */
```

Anche lo stesso callback di evento può essere utilizzato su un Widget con `user_data` diversi. Ad esempio:

```
lv_obj_add_event_cb(widget, increment_on_click, LV_EVENT_CLICKED, &num1);
lv_obj_add_event_cb(widget, increment_on_click, LV_EVENT_CLICKED, &num2);
```

Gli eventi verranno richiamati nell'ordine in cui sono stati aggiunti.

Altri Widget possono utilizzare lo stesso *callback di evento*.

Allo stesso modo, gli eventi possono essere associati a dispositivi di input e display in questo modo:

```
lv_display_add_event_cb(display, event_cb, LV_EVENT_RESOLUTION_CHANGED, NULL);
lv_indev_add_event_cb(indev, event_cb, LV_EVENT_CLICKED, NULL);
```

## 5.5.2 Rimozione di Eventi dai Widget

```
uint32_t i;
uint32_t event_cnt = lv_obj_get_event_count(widget);
for(i = 0; i < event_cnt; i++) {
    lv_event_desc_t *event_desc = lv_obj_get_event_desc(widget, i);
    if(lv_event_desc_get_cb(event_desc) == some_event_cb) {
        lv_obj_remove_event(widget, i);
        break;
    }
}
```

## 5.5.3 Codici Evento

I codici evento possono essere raggruppati in queste categorie: - Eventi dispositivo di input - Eventi disegno - Altri eventi - Eventi speciali - Eventi personalizzati

Tutti i Widget (come Buttons/Labels/Sliders ecc.), indipendentemente dal tipo, ricevono gli eventi *Input device*, *Drawing* e *Other*.

Tuttavia, gli *Special events* sono specifici di un particolare tipo di widget. Consultare la [documentazione dei widget](#) per sapere quando vengono inviati.

Gli *eventi personalizzati* vengono aggiunti dall'utente e non vengono mai inviati da LVGL.

Esistono i seguenti codici evento:

### Eventi del Dispositivo di Input

- `LV_EVENT_PRESSED`: Il Widget è stato premuto
- `LV_EVENT_PRESSING`: Il Widget è premuto (chiamato continuamente durante la pressione)
- `LV_EVENT_PRESS_LOST`: Il Widget è ancora premuto, ma il cursore/dito è scivolato fuori dal Widget

- ***LV\_EVENT\_SHORT\_CLICKED***: Il Widget è stato premuto per un breve periodo di tempo e poi rilasciato senza scorrere.
- ***LV\_EVENT\_SINGLE\_CLICKED***: Il Widget è stato premuto per un breve periodo di tempo e poi rilasciato senza scorrere, per la prima volta in una serie di clic. Una serie di clic si riferisce a più clic brevi in un breve periodo di tempo e a una distanza ridotta.
- ***LV\_EVENT\_DOUBLE\_CLICKED***: Il Widget è stato premuto per un breve periodo di tempo e poi rilasciato senza scorrere, per la seconda volta in una serie di clic.
- ***LV\_EVENT\_TRIPLE\_CLICKED***: Il Widget è stato premuto per un breve periodo di tempo e poi rilasciato senza scorrere, per la terza volta in una serie di clic.
- ***LV\_EVENT\_LONG\_PRESSED***: Il è stato premuto per almeno `long_press_time`. Non richiamato se si scorre.
- ***LV\_EVENT\_LONG\_PRESSED\_REPEAT***: Richiamato dopo `long_press_time` ogni `long_press_repeat_time` ms. Non richiamato se si scorre.
- ***LV\_EVENT\_CLICKED***: Richiamato al rilascio se non si scorre (indipendentemente dalla pressione prolungata)
- ***LV\_EVENT\_RELEASED***: Richiamato ogni volta che il Widget viene rilasciato
- ***LV\_EVENT\_SCROLL\_BEGIN***: Inizia lo Inizia lo scorrimento [scorrimento]. Il parametro evento è un puntatore all'animazione dello "scroll". Può essere modificato
- ***LV\_EVENT\_SCROLL\_THROW\_BEGIN***:
- ***LV\_EVENT\_SCROLL\_END***: Termina lo "scrolling"
- ***LV\_EVENT\_SCROLL***: Scrolling
- ***LV\_EVENT\_GESTURE***: È stato rilevato un gesto. Il gesto si ottiene con `lv_indev_get_gesture_dir(lv_indev_active())`
- ***LV\_EVENT\_KEY***: Una tasto viene inviato al Widget. Get the key with `lv_indev_get_key(lv_indev_active())`
- ***LV\_EVENT\_FOCUSED***: Il Widget ha ricevuto il focus
- ***LV\_EVENT\_DIFOCUSED***: Il Widget ha perso il focus
- ***LV\_EVENT\_LEAVE***: Il Widget ha perso il focus ma è ancora selezionato
- ***LV\_EVENT\_HIT\_TEST***: Esegue un hit-test avanzato
- ***LV\_EVENT\_INDEV\_RESET***: Indev [Input device] è stato resettato
- ***LV\_EVENT\_HOVER\_OVER***: Passaggio dell'Indev sul Widget
- ***LV\_EVENT\_HOVER\_LEAVE***: L'Indev lascia il Widget

## Eventi di Disegno

- ***LV\_EVENT\_COVER\_CHECK***: Verifica se il Widget copre completamente un'area. Il parametro evento è `lv_cover_check_info_t *`.
- ***LV\_EVENT\_REFR\_EXT\_DRAW\_SIZE***: Ottiene l'area di disegno aggiuntiva richiesta attorno al Widget (ad esempio per l'ombra). Il parametro evento è `int32_t *` per memorizzare le dimensioni.
- ***LV\_EVENT\_DRAW\_MAIN\_BEGIN***: Avvio della fase di disegno principale
- ***LV\_EVENT\_DRAW\_MAIN***: Esecuzione del disegno principale
- ***LV\_EVENT\_DRAW\_MAIN\_END***: Termine della fase di disegno principale
- ***LV\_EVENT\_DRAW\_POST\_BEGIN***: Avvio della fase di post-disegno (quando tutti i figli sono stati disegnati)
- ***LV\_EVENT\_DRAW\_POST***: Esecuzione della fase di post-disegno (quando tutti i figli sono stati disegnati)
- ***LV\_EVENT\_DRAW\_POST\_END***: Termine della fase di post-disegno (quando tutti i figli sono stati disegnati)

- *LV\_EVENT\_DRAW\_TASK\_ADDED*: Aggiunta di un task di disegno

### Attenzione

Nelle funzioni di callback degli eventi di disegno, la sequenza di rendering è già iniziata e, durante questo periodo, è vietato apportare modifiche agli attributi o agli stili di qualsiasi widget o creare/eliminare widget. Se si tenta di farlo, LVGL genererà un errore di asserzione con un messaggio che indica che l'invalidazione di un'area non è consentita durante il rendering.

## Eventi Speciali

- *LV\_EVENT\_VALUE\_CHANGED*: Il valore del Widget è cambiato (ad esempio, il cursore è stato spostato)
- *LV\_EVENT\_INSERT*: Un testo è stato inserito nel Widget. I dati dell'evento sono `char *` in fase di inserimento.
- *LV\_EVENT\_REFRESH*: Notifica al Widget di aggiornare qualcosa (per l'utente)
- *LV\_EVENT\_READY*: Un processo è terminato
- *LV\_EVENT\_CANCEL*: Un processo è stato annullato

## Altri Eventi

- *LV\_EVENT\_CREATE*: Il Widget è in fase di creazione
- *LV\_EVENT\_DELETE*: Il Widget è in fase di eliminazione
- *LV\_EVENT\_CHILD\_CHANGED*: Il figlio è stato rimosso, aggiunto o le sue dimensioni e posizione sono state modificate.
- *LV\_EVENT\_CHILD\_CREATED*: Il figlio è stato creato, viene sempre inoltrato a tutti i genitori.
- *LV\_EVENT\_CHILD\_DELETED*: Il figlio è stato eliminato, viene sempre inoltrato a tutti i genitori.
- *LV\_EVENT\_SCREEN\_UNLOAD\_START*: È stato avviato lo scaricamento di una schermata, attivato immediatamente alla chiamata di `scr_load()`.
- *LV\_EVENT\_SCREEN\_LOAD\_START*: È stato avviato il caricamento di una schermata, attivato allo scadere del ritardo per il cambio schermata.
- *LV\_EVENT\_SCREEN\_LOADED*: È stata caricata una schermata
- *LV\_EVENT\_SCREEN\_UNLOADED*: È stata scaricata una schermata
- *LV\_EVENT\_SIZE\_CHANGED*: Le coordinate/dimensioni del Widget sono cambiate
- *LV\_EVENT\_STYLE\_CHANGED*: Lo stile del Widget è cambiato
- *LV\_EVENT\_LAYOUT\_CHANGED*: La posizione degli elementi figlio è cambiata a causa di un ricalcolo del layout
- *LV\_EVENT\_GET\_SELF\_SIZE*: Ottiene la dimensione interna di un widget

## Eventi del Display

- *LV\_EVENT\_INVALIDATE\_AREA*: Un'area viene invalidata (contrassegnata per il ridisegno). `lv_event_get_param(e)` restituisce un puntatore a un oggetto `lv_area_t` con le coordinate dell'area invalidata. L'area può essere modificata liberamente se necessario per adattarla a un requisito specifico del display. Di solito necessario con i display monocromatici per invalidare  $N \times 8$  righe o colonne in un unico passaggio.
- *LV\_EVENT\_RESOLUTION\_CHANGED*: Inviato quando la risoluzione cambia a causa di `lv_display_set_resolution()` o di `lv_display_set_rotation()`.
- *LV\_EVENT\_COLOR\_FORMAT\_CHANGED*: Inviato come risultato di qualsiasi chiamata a `lv_display_set_color_format()`.

- *LV\_EVENT\_REFR\_REQUEST*: Inviato quando si è verificato un evento che richiede un ridisegno.
- *LV\_EVENT\_REFR\_START*: Inviato prima dell'inizio di un ciclo di aggiornamento. Inviato anche se non c'è nulla da ridisegnare.
- *LV\_EVENT\_REFR\_READY*: Inviato al completamento dell'aggiornamento (dopo il rendering e la chiamata a *Flush Callback*). Inviato anche se non si è verificato alcun ridisegno.
- *LV\_EVENT\_RENDER\_START*: Inviato appena prima dell'inizio del rendering.
- *LV\_EVENT\_RENDER\_READY*: Inviato dopo il completamento del rendering (prima di chiamare *Flush Callback*)
- *LV\_EVENT\_FLUSH\_START*: Inviato prima della chiamata a *Flush Callback*.
- *LV\_EVENT\_FLUSH\_FINISH*: Inviato dopo il ritorno della chiamata a *Flush Callback*.
- *LV\_EVENT\_FLUSH\_WAIT\_START*: Inviato all'inizio della chiamata interna a *wait\_for\_flushing()*  
-- si verifica indipendentemente dal fatto che si verifichi effettivamente un'attesa. La chiamata termina immediatamente se *disp->flushing == 0*.
- *LV\_EVENT\_FLUSH\_WAIT\_FINISH*: Inviato quando la chiamata a *wait\_for\_flushing()* sta per terminare, indipendentemente dal fatto che si sia verificata un'attesa effettiva.

## Eventi Personalizzati

È possibile registrare un numero qualsiasi di codici evento personalizzati tramite `uint32_t MY_EVENT_1 = lv_event_register_id()`

Possono essere inviati a qualsiasi widget con `lv_obj_send_event(widget, MY_EVENT_1, &some_data)`

## Evento del Refresh

*LV\_EVENT\_REFRESH* è un evento speciale perché è progettato per consentire all'utente di notificare a un widget di aggiornarsi. Alcuni esempi:

- notificare a un'etichetta di aggiornare il testo in base a una o più variabili (ad esempio, l'ora corrente)
- aggiornare un'etichetta quando cambia la lingua
- abilitare un pulsante se vengono soddisfatte determinate condizioni (ad esempio, viene inserito il PIN corretto)
- aggiungere/rimuovere stili da/a un widget se viene superato un limite, ecc.

### 5.5.4 Invio Manuale di Eventi

Per inviare manualmente eventi a un widget, utilizzare `lv_obj_send_event(widget, <EVENT_CODE>, &some_data)`.

Ad esempio, questo può essere utilizzato per chiudere manualmente una finestra di messaggio simulando la pressione di un pulsante (anche se esistono modi più semplici per farlo):

```
/* Simulate the press of the first button (indexes start from zero) */
uint32_t btn_id = 0;
lv_obj_send_event(mbox, LV_EVENT_VALUE_CHANGED, &btn_id);
```

Lo stesso vale per i dispositivi di visualizzazione e input con `lv_display_send_event(widget, <EVENT_CODE>, &some_data)` e `lv_indev_send_event(widget, <EVENT_CODE>, &some_data)`.

### 5.5.5 Campi di `lv_event_t`

`lv_event_t` è l'unico parametro passato al callback dell'evento e contiene tutti i dati relativi all'evento. Da esso si possono ottenere i seguenti valori:

- `lv_event_get_code(e)`: ottiene il codice dell'evento

- `lv_event_get_current_target(e)`: ottiene il Widget a cui è stato inviato un evento. Ovvero il Widget il cui gestore eventi viene chiamato.
- `lv_event_get_target(e)`: ottiene il Widget che ha originariamente attivato l'evento (diverso da `lv_event_get_target()` se *event bubbling* è abilitato)
- `lv_event_get_user_data(e)`: ottiene il puntatore passato come ultimo parametro di `lv_obj_add_event_cb()`.
- `lv_event_get_param(e)`: ottiene il parametro passato come ultimo parametro di `lv_obj_send_event_cb()`

## 5.5.6 Event Bubbling

If `lv_obj_add_flag(widget, LV_OBJ_FLAG_EVENT_BUBBLE)` is enabled all events will be sent to a Widget's parent as well. If the parent also has `LV_OBJ_FLAG_EVENT_BUBBLE` enabled the event will be sent to its parent, and so on.

The *target* parameter of the event is always the current target Widget, not the original Widget. To get the original target call `lv_event_get_target_obj(e)` in the event handler.

## 5.5.7 Event Trickle

Also known as Event Capturing, if `lv_obj_add_flag(widget, LV_OBJ_FLAG_EVENT_TRICKLE)` is enabled all events will be sent to the Widget's children as well. This is the opposite of event bubbling --- instead of propagating up the parent, events propagate down to the children.

The trickle mechanism only affects immediate children, not grandchildren or deeper descendants. If you need events to propagate to deeper levels, each child would need to have the `LV_OBJ_FLAG_EVENT_TRICKLE` flag enabled.

Like with bubbling, the *target* parameter of the event is always the current target Widget, not the original Widget. To get the original target call `lv_event_get_target_obj(e)` in the event handler.

## 5.5.8 Esempi

### Evento clic pulsante

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    LV_LOG_USER("Clicked");

    static uint32_t cnt = 1;
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    lv_label_set_text_fmt(label, "%" LV_PRIu32, cnt);
    cnt++;
}

/**
 * Add click event to a button
 */
void lv_example_event_click(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);

    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Click me!");
    lv_obj_center(label);
}

#endif
```

### Sequenze di clic

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

static void short_click_event_cb(lv_event_t * e)
{
    LV_LOG_USER("Short clicked");

    lv_obj_t * info_label = (lv_obj_t *) lv_event_get_user_data(e);
    lv_indev_t * index = (lv_indev_t *) lv_event_get_param(e);
    uint8_t cnt = lv_indev_get_short_click_streak(index);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_label_set_text_fmt(info_label, "Short click streak: %u", cnt);
}

static void streak_event_cb(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    const char * text = (const char *) lv_event_get_user_data(e);
    lv_label_set_text(label, text);
}

void lv_example_event_streak(void)
{
    lv_obj_t * info_label = lv_label_create(lv_screen_active());
    lv_label_set_text(info_label, "No events yet");

    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);
    lv_obj_add_event_cb(btn, short_click_event_cb, LV_EVENT_SHORT_CLICKED, info_label);
    lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_SINGLE_CLICKED, (void *) "Single clicked");
    lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_DOUBLE_CLICKED, (void *) "Double clicked");
    lv_obj_add_event_cb(btn, streak_event_cb, LV_EVENT_TRIPLE_CLICKED, (void *) "Triple clicked");

    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Click me!");
    lv_obj_center(label);
}
#endif

```

## Gestione di eventi multipli

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * label = (lv_obj_t *) lv_event_get_user_data(e);

    switch(code) {
        case LV_EVENT_PRESSED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_PRESSED");
            break;
        case LV_EVENT_CLICKED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_CLICKED");
            break;
        case LV_EVENT_LONG_PRESSED:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED");
            break;
        case LV_EVENT_LONG_PRESSED_REPEAT:
            lv_label_set_text(label, "The last button event:\nLV_EVENT_LONG_PRESSED_REPEAT");
            break;
        default:
            break;
    }
}

/**
 * Handle multiple events
 */
void lv_example_event_button(void)
{
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, 100, 50);
    lv_obj_center(btn);

    lv_obj_t * btn_label = lv_label_create(btn);
    lv_label_set_text(btn_label, "Click me!");
    lv_obj_center(btn_label);

    lv_obj_t * info_label = lv_label_create(lv_screen_active());
    lv_label_set_text(info_label, "The last button event:\nNone");

    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_ALL, info_label);
}
#endif

```

## Evento bubbling

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void event_cb(lv_event_t * e)
{
    /*The original target of the event. Can be the buttons or the container*/
    lv_obj_t * target = lv_event_get_target_obj(e);

    /*The current target is always the container as the event is added to it*/
    lv_obj_t * cont = lv_event_get_current_target_obj(e);

    /*If container was clicked do nothing*/
    if(target == cont) return;

    /*Make the clicked buttons red*/
    lv_obj_set_style_bg_color(target, lv_palette_main(LV_PALETTE_RED), 0);
}

/**
 * Demonstrate event bubbling
 */
void lv_example_event_bubble(void)

```

(continues on next page)

(continua dalla pagina precedente)

```
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 290, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 30; i++) {
        lv_obj_t * btn = lv_button_create(cont);
        lv_obj_set_size(btn, 70, 50);
        lv_obj_add_flag(btn, LV_OBJ_FLAG_EVENT_BUBBLE);

        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_obj_add_event_cb(cont, event_cb, LV_EVENT_CLICKED, NULL);
}

#endif
```

## Event trickle-down

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

/***
 * Demonstrate event trickle
 */
void lv_example_event_trickle(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 290, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    static lv_style_t style_black;
    lv_style_init(&style_black);
    lv_style_set_text_color(&style_black, lv_color_white());
    lv_style_set_bg_color(&style_black, lv_color_black());

    /*Enable event trickle-down on the container*/
    lv_obj_add_flag(cont, LV_OBJ_FLAG_EVENT_TRICKLE);

    lv_obj_add_style(cont, &style_black, LV_STATE_PRESSED);

    uint32_t i;
    for(i = 0; i < 9; i++) {
        lv_obj_t * subcont = lv_obj_create(cont);
        lv_obj_set_size(subcont, 70, 50);
        lv_obj_t * label = lv_label_create(subcont);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);

        /*Add style to the label when clicked*/
        lv_obj_add_style(subcont, &style_black, LV_STATE_FOCUSED);
    }
}
#endif
```

## Evento Draw

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

static int32_t size = 0;
static bool size_dec = false;

static void timer_cb(lv_timer_t * timer)
{
    lv_obj_invalidate((lv_obj_t *) lv_timer_get_user_data(timer));
    if(size_dec) size--;
    else size++;

    if(size == 50) size_dec = true;
    else if(size == 0) size_dec = false;
}

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *) lv_draw_task_get_draw_dsc(draw_task);
    if(base_dsc->part == LV_PART_MAIN) {
        lv_draw_rect_dsc_t draw_dsc;
        lv_draw_rect_dsc_init(&draw_dsc);
        draw_dsc.bg_color = LV_RADIUS_CIRCLE;
        draw_dsc.radius = LV_RADIUS_CIRCLE;
        draw_dsc.border_color = lv_color_hex(0xffff5555);
        draw_dsc.border_width = 2;
        draw_dsc.outline_color = lv_color_hex(0xff0000);
        draw_dsc.outline_pad = 3;
        draw_dsc.outline_width = 2;

        lv_area_t a;
        a.x1 = 0;
        a.y1 = 0;
        a.x2 = size;
        a.y2 = size;
        lv_area_t obj_coords;
        lv_obj_get_coords(obj, &obj_coords);
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_area_align(&obj_coords, &a, LV_ALIGN_CENTER, 0, 0);
    lv_draw_rect(base_dsc->layer, &draw_dsc, &a);
}

/**
 * Demonstrate the usage of draw event
 */
void lv_example_event_draw(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 200, 200);
    lv_obj_center(cont);
    lv_obj_add_event_cb(cont, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(cont, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_timer_create(timer_cb, 30, cont);
}

#endif

```

## 5.5.9 API

*lv\_obj.h*  
*lv\_obj\_event.h*  
*lv\_obj\_event\_private.h*  
*lv\_display.h*  
*lv\_indev.h*  
*lv\_event.h*  
*lv\_event\_private.h*

## 5.6 Layout

### 5.6.1 Flex

#### Panoramica

Flexbox (o Flex in breve) è un sottoinsieme dei comportamenti del CSS Flexbox.

Si possono disporre gli elementi (Widget figli) in righe o colonne (track), gestire il wrapping, regolare la spaziatura tra elementi e track, gestire l'aumento di dimensione per far sì che gli elementi riempiano lo spazio rimanente rispetto alla larghezza e all'altezza minime/massime.

Per trasformare un Widget in un contenitore Flex, si chiama `lv_obj_set_layout(widget, LV_LAYOUT_FLEX)`.

Si noti che la funzionalità di layout Flex di LVGL deve essere abilitata globalmente con `LV_USE_FLEX` in `lv_conf.h`.

#### Termini

- **track** [tracce]: righe o colonne
- **direzione principale**: riga o colonna, la direzione in cui vengono posizionati per primi più elementi
- **direzione trasversale**: la direzione perpendicolare alla **direzione principale**
- **wrap**: se non c'è più spazio nella traccia, ne viene iniziata una nuova
- **grow** [crescita]: se impostato su un elemento, "crescerà" per riempire lo spazio rimanente nella traccia Lo spazio disponibile verrà distribuito tra gli elementi in base al loro valore di crescita (un valore maggiore significa più spazio)
- **gap**: lo spazio tra righe e colonne o tra gli elementi di una track

Vedere [CSS Flexbox](#) per illustrazioni che mostrano il significato di questi termini.

## Interfaccia Semplice

Si usano le seguenti funzioni per impostare e controllare il layout Flex su qualsiasi Widget genitore.

### Nota

Affinché questi stili abbiano effetto, il Widget genitore deve essere un contenitore Flex. Le funzioni seguenti fanno sì che il Widget genitore diventi un contenitore Flex, se non lo è già.

## Flusso Flex

`lv_obj_set_flex_flow(widget, flex_flow)`

I possibili valori per `flex_flow` sono:

- `LV_FLEX_FLOW_ROW`: Posiziona gli elementi figlio in una riga senza riavvolgimento
- `LV_FLEX_FLOW_COLUMN`: Posiziona gli elementi figlio in una colonna senza riavvolgimento
- `LV_FLEX_FLOW_ROW_WRAP`: Posiziona gli elementi figlio in una riga con riavvolgimento
- `LV_FLEX_FLOW_COLUMN_WRAP`: Posiziona gli elementi figlio in una colonna con riavvolgimento
- `LV_FLEX_FLOW_ROW_REVERSE`: Posiziona gli elementi figlio in una riga senza riavvolgimento ma in ordine inverso
- `LV_FLEX_FLOW_COLUMN_REVERSE`: Posiziona gli elementi figlio in una colonna senza riavvolgimento ma in ordine inverso
- `LV_FLEX_FLOW_ROW_WRAP_REVERSE`: Posiziona gli elementi figlio in una riga con riavvolgimento ma in ordine inverso
- `LV_FLEX_FLOW_COLUMN_WRAP_REVERSE`: Posiziona gli elementi figlio in una colonna con riavvolgimento ma in ordine inverso

Questi valori fanno sì che il comportamento del layout del Widget modelli il comportamento di CSS Flexbox combinando `flex-direction` e `flex-wrap` come definito in `flex-flow`.

## Allineamento Flex

Per gestire il posizionamento degli elementi figlio, si usa `lv_obj_set_flex_align(widget, main_place, cross_place, track_cross_place)` che fa sì che il modello del widget padre imposta il comportamento del contenitore Flex definito [qui](#).

- `main_place` determina come distribuire gli elementi nella loro track sull'asse principale. Ad esempio, allinea gli elementi a destra su `LV_FLEX_FLOW_ROW_WRAP`. (In CSS si chiama `justify-content`.)
- `cross_place` determina come distribuire gli elementi nella loro track sull'asse trasversale. Ad esempio, se gli elementi hanno altezze diverse, li allinea alla parte inferiore della track. (In CSS si chiama `align-items`.)
- `track_cross_place` determina come distribuire le track (in CSS si chiama `align-content`.)

I valori possibili sono:

- `LV_FLEX_ALIGN_START`: indica sinistra quando la direzione è orizzontale, in alto quando è verticale (il default)
- `LV_FLEX_ALIGN_END`: indica destra quando la direzione è orizzontale, in basso quando è verticale
- `LV_FLEX_ALIGN_CENTER`: semplicemente centra rispetto alla direzione
- `LV_FLEX_ALIGN_SPACE_EVENLY`: gli elementi vengono distribuiti in modo che la spaziatura tra due elementi qualsiasi (e lo spazio tra i bordi) sia uguale. Non si applica a `track_cross_place`.
- `LV_FLEX_ALIGN_SPACE_AROUND`: gli elementi sono distribuiti uniformemente nella track con uguale spazio attorno a loro. Si noti che visivamente gli spazi non sono uguali poiché tutti gli elementi hanno uguale

spazio su entrambi i lati. Questo fa sì che lo spazio tra gli elementi sia il doppio dello spazio tra gli elementi sul bordo e il bordo del contenitore. Non si applica a `track_cross_place`.

- `LV_FLEX_ALIGN_SPACE_BETWEEN`: gli elementi sono distribuiti uniformemente nella traccia senza spazio prima e dopo il primo e l'ultimo elemento. Non si applica a `track_cross_place`.

Vedere `justify-content`, `align-items` and `align-content` per le illustrazioni di questi valori.

## Flex grow

Flex grow [crescita] può essere utilizzato per far sì che uno o più widget figlio occupino lo spazio disponibile nella track. Quando più di un widget figlio ha valori di grow diversi da zero, tutto lo spazio disponibile verrà distribuito in proporzione ai rispettivi valori di grow. Ad esempio, se ci sono 400 px di spazio rimanenti e 3 widget figlio con valori di crescita diversi da zero:

- A con grow = 1
- B con grow = 1
- C con grow = 2

A e B occuperanno 100 px, e C occuperà 200 px.

Flex grow può essere impostato su un Widget figlio con `lv_obj_set_flex_grow(child, value)`. `value` deve essere  $\geq 1$  o 0 per disabilitare la crescita sul widget figlio.

Vedere `flex-grow` per un'illustrazione di questo comportamento.

## Interfaccia dello Stile

Tutti i valori relativi a Flex sono proprietà di stile nascoste, quindi si possono usare come qualsiasi altra proprietà di stile.

Esistono le seguenti proprietà di stile relative a Flex:

- `FLEX_FLOW`
- `FLEX_MAIN_PLACE`
- `FLEX_CROSS_PLACE`
- `FLEX_TRACK_PLACE`
- `FLEX_GROW`

## Spaziatura interna

Per modificare lo spazio minimo inserito da Flexbox tra i Widget, è possibile utilizzare le seguenti funzioni per impostare lo stile della spaziatura del contenitore Flex:

- `lv_style_set_pad_row()` imposta la spaziatura interna tra le righe.
- `lv_style_set_pad_column()` imposta la spaziatura interna tra le colonne.

Possono essere utilizzati, ad esempio, se non si desidera alcun padding tra i Widgets: `lv_style_set_pad_column(&row_container_style, 0)`

## Altre Funzionalità

### RTL

Se la direzione di base del contenitore è impostata su `LV_BASE_DIRRTL` il significato di `LV_FLEX_ALIGN_START` e `LV_FLEX_ALIGN_END` viene scambiato nei layout ROW. Ad esempio, START significherà destra.

Gli elementi nei layout ROW e le track di COLUMN verranno posizionati da destra a sinistra.

## Nuova track

È possibile forzare Flex a inserire un elemento in una nuova riga con `lv_obj_add_flag(child, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK)`.

### Ulteriori Informazioni

Maggiori informazioni su CSS Flexbox.

## Esempi

### Un semplice layout di riga e colonna con flexbox

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * A simple row and a column layout with flexbox
 */
void lv_example_flex_1(void)
{
    /*Create a container with ROW flex direction*/
    lv_obj_t * cont_row = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_row, 300, 75);
    lv_obj_align(cont_row, LV_ALIGN_TOP_MID, 0, 5);
    lv_obj_set-flex_flow(cont_row, LV_FLEX_FLOW_ROW);

    /*Create a container with COLUMN flex direction*/
    lv_obj_t * cont_col = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_col, 200, 150);
    lv_obj_align_to(cont_col, cont_row, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
    lv_obj_set-flex_flow(cont_col, LV_FLEX_FLOW_COLUMN);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * obj;
        lv_obj_t * label;

        /*Add items to the row*/
        obj = lv_button_create(cont_row);
        lv_obj_set_size(obj, 100, LV_PCT(100));

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %" LV_PRIu32 "", i);
        lv_obj_center(label);

        /*Add items to the column*/
        obj = lv_button_create(cont_col);
        lv_obj_set_size(obj, LV_PCT(100), LV_SIZE_CONTENT);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %" LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

### Disporre gli elementi in righe con avvolgimento e spaziatura uniforme

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * Arrange items in rows with wrap and place the items to get even space around them.
 */
void lv_example_flex_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set-flex_flow(&style, LV_FLEX_FLOW_ROW_WRAP);
    lv_style_set-flex_main_place(&style, LV_FLEX_ALIGN_SPACE_EVENLY);
    lv_style_set-layout(&style, LV_LAYOUT_FLEX);

    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_add_style(cont, &style, 0);

    uint32_t i;
    for(i = 0; i < 8; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);
        lv_obj_add_flag(obj, LV_OBJ_FLAG_CHECKABLE);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

## Dimostrare la flex grow.

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * Demonstrate flex grow.
 */
void lv_example_flex_3(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set.flex_flow(cont, LV_FLEX_FLOW_ROW);

    lv_obj_t * obj;
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, 40, 40);           /*Fix size*/

    obj = lv_obj_create(cont);
    lv_obj_set_height(obj, 40);
    lv_obj_set.flex_grow(obj, 1);          /*1 portion from the free space*/

    obj = lv_obj_create(cont);
    lv_obj_set_height(obj, 40);
    lv_obj_set.flex_grow(obj, 2);          /*2 portion from the free space*/

    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, 40, 40);           /*Fix size. It is flushed to the right by the "grow" items*/
}

#endif
```

## Dimostrare la flex grow.

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * Reverse the order of flex items
 */
void lv_example_flex_4(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set.flex_flow(cont, LV_FLEX_FLOW_COLUMN_REVERSE);

    uint32_t i;
    for(i = 0; i < 6; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 100, 50);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item: %" LV_PRIu32, i);
        lv_obj_center(label);
    }
}

#endif
```

## Dimostrare le proprietà di stile degli spazi tra colonne e righe

```
#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row((lv_obj_t *) obj, v, 0);
}

static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column((lv_obj_t *) obj, v, 0);
}

/***
 * Demonstrate the effect of column and row gap style properties
 */
void lv_example_flex_5(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set.flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 9; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, cont);
    lv_anim_set_values(&a, 0, 10);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_set_exec_cb(&a, row_gap_anim);
lv_anim_set_duration(&a, 500);
lv_anim_set_reverse_duration(&a, 500);
lv_anim_start(&a);

lv_anim_set_exec_cb(&a, column_gap_anim);
lv_anim_set_duration(&a, 3000);
lv_anim_set_reverse_duration(&a, 3000);
lv_anim_start(&a);
}

#endif

```

## La direzione di base RTL modifica l'ordine degli elementi

```

#include "../../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

/**
 * RTL base direction changes order of the items.
 * Also demonstrate how horizontal scrolling works with RTL.
 */
void lv_example_flex_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_style_base_dir(cont, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW_WRAP);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * obj = lv_obj_create(cont);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);

        lv_obj_t * label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32, i);
        lv_obj_center(label);
    }
}
#endif

```

## API

*lv\_flex.h*

### 5.6.2 Grid

#### Panoramica

Il layout "Grid" [a griglia] è un sottoinsieme del layout CSS Grid.

Può disporre gli elementi (Widget figlio) in una "tabella" 2D con righe e colonne (track). Un elemento può estendersi su più colonne o righe. La dimensione della track può essere impostata in pixel, in base all'elemento più grande ([LV\\_GRID\\_CONTENT](#)) o in base a una frazione dello spazio libero disponibile (ad esempio [Free \[FR\] Units](#)) per distribuire lo spazio libero in modo proporzionale.

Per trasformare un Widget in un contenitore Grid, chiamare [\*lv\\_obj\\_set\\_layout\*\(widget, LV\\_LAYOUT\\_GRID\)](#).

Si noti che la funzionalità di layout Grid di LVGL deve essere abilitata globalmente con [LV\\_USE\\_GRID](#) in [lv\\_conf.h](#).

#### Termini

- **track** [tracce]: righe o colonne
- **free (FR) units**: se la dimensione di una track è impostata in **unità FR**, questa aumenterà fino a riempire lo spazio rimanente nel Widget padre (contenitore), in proporzione alle altre tracce che hanno valori di unità FR diversi da zero.
- **gap**: lo spazio tra righe e colonne o tra gli elementi di una track

## Interfaccia Semplice

Con le seguenti funzioni è possibile impostare qualsiasi Widget padre su un layout Grid.

### Nota

Come per i contenitori Flex, il Widget padre deve essere un contenitore Grid affinché questi stili abbiano effetto. Le funzioni seguenti fanno sì che il Widget padre diventi un contenitore Grid, se non lo è già.

## Descrittori di Grid

Per prima cosa è necessario descrivere la dimensione di righe e colonne. Questo può essere fatto dichiarando 2 array e le dimensioni delle track in essi contenute. L'ultimo elemento deve essere `LV_GRID_TEMPLATE_LAST`.

Ad esempio:

```
static int32_t column_dsc[] = {100, 400, LV_GRID_TEMPLATE_LAST}; /* 2 columns with 100- and 400-px width */
static int32_t row_dsc[] = {100, 100, 100, LV_GRID_TEMPLATE_LAST}; /* 3 100-px tall rows */
```

Per impostare i descrittori su un elemento padre, si usa `lv_obj_set_grid_dsc_array(widget, col_dsc, row_dsc)`.

Oltre a impostare le dimensioni in pixel, è possibile utilizzare due valori speciali:

- `LV_GRID_CONTENT` imposta la dimensione in modo che si adatti all'elemento figlio più grande di questa track
- `LV_GRID_FR(X)` determina quale porzione dello spazio rimanente deve essere utilizzata da questa track. Valori più grandi indicano spazio maggiore.

## Elementi della Grid

Per default, gli elementi figlio di un Widget non vengono aggiunti alla grid. Devono essere aggiunti manualmente a una cella.

Per farlo, si chiama `lv_obj_set_grid_cell(child, column_align, column_pos, column_span, row_align, row_pos, row_span)`.

`column_align` e `row_align` determinano come allineare il Widget figlio nella sua cella. I valori possibili sono:

- `LV_GRID_ALIGN_START`: indica a sinistra quando la direzione è orizzontale e in alto quando è verticale (il default)
- `LV_GRID_ALIGN_END`: indica a destra quando la direzione è orizzontale e in basso quando è verticale
- `LV_GRID_ALIGN_CENTER`: semplicemente centra `column_pos` e `row_pos` indica l'indice a partire da zero della cella in cui deve essere posizionato l'elemento.

`column_span` e `row_span` indicano quante track devono essere occupate dalla cella iniziale. Deve essere  $\geq 1$ .

## Allineamento nella Grid

Se c'è dello spazio vuoto, gli elementi (Widget) nelle track della Grid possono essere allineati in diversi modi:

- `LV_GRID_ALIGN_START`: indica sinistra quando la direzione è orizzontale e in alto quando è verticale. (il default)
- `LV_GRID_ALIGN_END`: indica a destra quando la direzione è orizzontale e in basso quando è verticale
- `LV_GRID_ALIGN_CENTER`: semplicemente centra
- `LV_GRID_ALIGN_SPACE_EVENLY`: gli elementi sono distribuiti in modo che la spaziatura tra due elementi qualsiasi (e lo spazio fino ai bordi) sia uguale. Non si applica a `track_cross_place`.

- ***LV\_GRID\_ALIGN\_SPACE\_AROUND***: gli elementi sono distribuiti uniformemente nella track con lo stesso spazio intorno a loro. Si noti che visivamente gli spazi non sono uguali, poiché tutti gli elementi hanno lo stesso spazio su entrambi i lati. Questo fa sì che lo spazio tra gli elementi sia il doppio dello spazio tra gli elementi sul bordo e il bordo del contenitore. Non si applica a `track_cross_place`.
- ***LV\_GRID\_ALIGN\_SPACE\_BETWEEN***: gli elementi sono distribuiti uniformemente nella track con il primo e l'ultimo elemento adiacenti ai bordi del contenitore. Non si applica a `track_cross_place`.

Per impostare l'allineamento della track, si usa `lv_obj_set_grid_align(widget, column_align, row_align)`.

### Sub grid

Se si impostano i descrittori di grid di colonna e/o riga di un widget su `NULL`, verranno utilizzati i descrittori di grid del genitore.

Ad esempio, se si crea un elemento grid che si estende sulle colonne 2..6 e sulle righe 1..3 della grid, l'elemento grid occuperà 5 colonne e 4 righe con le dimensioni di track corrispondenti al contenitore Grid padre.

In questo modo, anche se nella grid viene utilizzato un elemento wrapper, può essere reso "trasparente" dal punto di vista della grid.

Limitazioni:

- La sub-grid viene risolta solo fino a una profondità di 1 livello. Ovvero, una grid può avere una sub-grid figlia, ma tale sub-grid non può avere un'altra sub-grid.
- ***LV\_GRID\_CONTENT*** le track sulla grid non vengono gestite nella sub-grid, ma solo nella propria grid.

La funzionalità della sub-grid funziona come in CSS. Per ulteriori informazioni, vedere [CSS Subgrid](#).

### Interfaccia dello Stile

Tutti i valori relativi alla Grid sono proprietà di stile nascoste, quindi è possibile utilizzarle come qualsiasi altra proprietà di stile.

Esistono le seguenti proprietà di stile relative alla Grid:

- `GRID_COLUMN_DSC_ARRAY`
- `GRID_ROW_DSC_ARRAY`
- `GRID_COLUMN_ALIGN`
- `GRID_ROW_ALIGN`
- `GRID_CELL_X_ALIGN`
- `GRID_CELL_COLUMN_POS`
- `GRID_CELL_COLUMN_SPAN`
- `GRID_CELL_Y_ALIGN`
- `GRID_CELL_ROW_POS`
- `GRID_CELL_ROW_SPAN`

### Spaziatura interna

Per modificare lo spazio minimo di inserimento della Grid tra i widget, è possibile impostare le seguenti proprietà nello stile del contenitore della Grid:

- `lv_style_set_pad_row()` impone la spaziatura interna tra le righe.
- `lv_style_set_pad_column()` impone la spaziatura interna tra le colonne.

## Altre funzionalità

### RTL

Se la direzione di base del contenitore è impostata su `LV_BASE_DIR_RTL`, il significato di `LV_GRID_ALIGN_START` e `LV_GRID_ALIGN_END` viene invertito. Ad esempio, START significherà "più a destra".

Le colonne verranno posizionate da destra a sinistra.

### Ulteriori Informazioni

- Maggiori informazioni sul layout [CSS Grid](#).
- Maggiori informazioni sul layout [CSS Subgrid](#).

## Esempio

### Una semplice grid

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * A simple grid
 */
void lv_example_grid_1(void)
{
    static int32_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_style_grid_column_dsc_array(cont, col_dsc, 0);
    lv_obj_set_style_grid_row_dsc_array(cont, row_dsc, 0);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_layout(cont, LV_LAYOUT_GRID);

    lv_obj_t * label;
    lv_obj_t * obj;

    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_button_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                             LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "c%d, r%d", col, row);
        lv_obj_center(label);
    }
}

#endif
```

### Posizionamento e ampiezza delle celle

```
#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/***
 * Demonstrate cell placement and span
 */
void lv_example_grid_2(void)
{
    static int32_t col_dsc[] = {70, 70, 70, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {50, 50, 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);

    lv_obj_t * label;
    lv_obj_t * obj;

    /*Cell to 0;0 and align to to the start (left/top) horizontally and vertically too*/
    obj = lv_obj_create(cont);
    lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 0, 1,
                         LV_GRID_ALIGN_START, 0, 1);
    label = lv_label_create(obj);
    lv_label_set_text(label, "c0, r0");

    /*Cell to 1;0 and align to to the start (left) horizontally and center vertically too*/
    obj = lv_obj_create(cont);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 1, 1,
                     LV_GRID_ALIGN_CENTER, 0, 1);
label = lv_label_create(obj);
lv_label_set_text(label, "c1, r0");

/*Cell to 2;0 and align to the start (left) horizontally and end (bottom) vertically too*/
obj = lv_obj_create(cont);
lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_START, 2, 1,
                     LV_GRID_ALIGN_END, 0, 1);
label = lv_label_create(obj);
lv_label_set_text(label, "c2, r0");

/*Cell to 1;1 but 2 column wide (span = 2).Set width and height to stretched.*/
obj = lv_obj_create(cont);
lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 1, 2,
                     LV_GRID_ALIGN_STRETCH, 1, 1);
label = lv_label_create(obj);
lv_label_set_text(label, "c1-2, r1");

/*Cell to 0;1 but 2 rows tall (span = 2).Set width and height to stretched.*/
obj = lv_obj_create(cont);
lv_obj_set_size(obj, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, 0, 1,
                     LV_GRID_ALIGN_STRETCH, 1, 2);
label = lv_label_create(obj);
lv_label_set_text(label, "c0\nr1-2");
}

#endif

```

## La -free unit- della grid

```

#include "../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate grid's "free unit"
 */
void lv_example_grid_3(void)
{
    /*Column 1: fix width 60 px
     *Column 2: 1 unit from the remaining free space
     *Column 3: 2 unit from the remaining free space*/
    static int32_t col_dsc[] = {60, LV_GRID_FR(1), LV_GRID_FR(2), LV_GRID_TEMPLATE_LAST};

    /*Row 1: fix width 50 px
     *Row 2: 1 unit from the remaining free space
     *Row 3: fix width 50 px*/
    static int32_t row_dsc[] = {50, LV_GRID_FR(1), 50, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        /*Stretch the cell horizontally and vertically too
         *Set span to 1 to make the cell 1 column/row sized*/
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                            LV_GRID_ALIGN_STRETCH, row, 1);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }
}
#endif

```

## Posizionamento delle tracce

```

#include "../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate track placement
 */
void lv_example_grid_4(void)
{
    static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Add space between the columns and move the rows to the bottom (end)*

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_grid_align(cont, LV_GRID_ALIGN_SPACE_BETWEEN, LV_GRID_ALIGN_END);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);

    lv_obj_t * label;

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * obj;
uint8_t i;
for(i = 0; i < 9; i++) {
    uint8_t col = i % 3;
    uint8_t row = i / 3;

    obj = lv_obj_create(cont);
    /*Stretch the cell horizontally and vertically too
     *Set span to 1 to make the cell 1 column/row sized*/
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                         LV_GRID_ALIGN_STRETCH, row, 1);

    label = lv_label_create(obj);
    lv_label_set_text_fmt(label, "%d,%d", col, row);
    lv_obj_center(label);
}
#endif

```

## Spazi tra colonne e righe

```

#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

static void row_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_row((lv_obj_t *) obj, v, 0);
}

static void column_gap_anim(void * obj, int32_t v)
{
    lv_obj_set_style_pad_column((lv_obj_t *) obj, v, 0);
}

/**
 * Demonstrate column and row gap
 */
void lv_example_grid_5(void)
{
    /*60x60 cells*/
    static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

    lv_obj_t * label;
    lv_obj_t * obj;
    uint8_t i;
    for(i = 0; i < 9; i++) {
        uint8_t col = i % 3;
        uint8_t row = i / 3;

        obj = lv_obj_create(cont);
        lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                            LV_GRID_ALIGN_STRETCH, row, 1);
        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%d,%d", col, row);
        lv_obj_center(label);
    }

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, cont);
    lv_anim_set_values(&a, 0, 10);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);

    lv_anim_set_exec_cb(&a, row_gap_anim);
    lv_anim_set_duration(&a, 500);
    lv_anim_set_reverse_duration(&a, 500);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, column_gap_anim);
    lv_anim_set_duration(&a, 3000);
    lv_anim_set_reverse_duration(&a, 3000);
    lv_anim_start(&a);
}

#endif

```

## Direzione RTL sulla griglia

```

#include "../../lv_examples.h"
#if LV_USE_GRID && LV_BUILD_EXAMPLES

/**
 * Demonstrate RTL direction on grid
 */
void lv_example_grid_6(void)
{
    static int32_t col_dsc[] = {60, 60, 60, LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {45, 45, 45, LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 300, 220);
    lv_obj_center(cont);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_style_base_dir(cont, LV_BASE_DIRRTL, 0);
lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);

lv_obj_t * label;
lv_obj_t * obj;
uint8_t i;
for(i = 0; i < 9; i++) {
    uint8_t col = i % 3;
    uint8_t row = i / 3;

    obj = lv_obj_create(cont);
    /*Stretch the cell horizontally and vertically too
     *Set span to 1 to make the cell 1 column/row sized*/
    lv_obj_set_grid_cell(obj, LV_GRID_ALIGN_STRETCH, col, 1,
                         LV_GRID_ALIGN_STRETCH, row, 1);

    label = lv_label_create(obj);
    lv_label_set_text_fmt(label, "%d,%d", col, row);
    lv_obj_center(label);
}

#endif
}

```

## API

*lv\_grid.h*

*lv\_gridnav.h*

## 5.7 Scrolling

### 5.7.1 Panoramica

In LVGL lo "scrolling" [scorrimento] funziona in modo molto intuitivo: se un Widget si trova al di fuori dell'area di contenuto principale (la dimensione senza padding), quest'ultima diventa scorrevole e vengono visualizzate le barre di scorrimento. Tutto qui.

Qualsiasi widget può essere "scrollable" [scorrevole], inclusi *Widget Base (lv\_obj)*, *lv\_image*, *lv\_button*, *lv\_meter*, ecc.

Il Widget può essere scorrevole orizzontalmente o verticalmente con un solo movimento; lo scorrimento diagonale non è possibile.

### Scrollbar

#### Modalità

Le barre di scorrimento vengono visualizzate in base alla `scrollbar-mode` configurata. Sono disponibili le seguenti modalità:

- *LV\_SCROLLBAR\_MODE\_OFF*: Non mostrare mai le barre di scorrimento
- *LV\_SCROLLBAR\_MODE\_ON*: Mostra sempre le barre di scorrimento
- *LV\_SCROLLBAR\_MODE\_ACTIVE*: Mostra le barre di scorrimento durante lo scorrimento di un Widget
- *LV\_SCROLLBAR\_MODE\_AUTO*: Mostra le barre di scorrimento quando il contenuto è sufficientemente grande da poter essere scorrevole

`lv_obj_set_scrollbar_mode(widget, LV_SCROLLBAR_MODE_...)` imposta la modalità della barra di scorrimento per un widget.

### Stili

Una "Scrollbar" [barra di scorrimento] è una parte dedicata di un widget, chiamata *LV\_PART\_SCROLLBAR*. Ad esempio, una barra di scorrimento può diventare rossa in questo modo:

```

static lv_style_t style_red;
lv_style_init(&style_red);
lv_style_set_bg_color(&style_red, lv_color_red());

...
lv_obj_add_style(widget, &style_red, LV_PART_SCROLLBAR);

```

Un Widget passa allo stato `LV_STATE_SCROLLLED` durante lo scorrimento. Questo consente di aggiungere diversi stili al Widget, che saranno attivi durante lo scorrimento. Ad esempio, questo codice rende la barra di scorrimento blu durante lo scorrimento del Widget:

```
static lv_style_t style_blue;
lv_style_init(&style_blue);
lv_style_set_bg_color(&style_blue, lv_color_blue());
...
lv_obj_add_style(widget, &style_blue, LV_STATE_SCROLLLED | LV_PART_SCROLLBAR);
```

Se la direzione di base di `LV_PART_SCROLLBAR` è RTL (`LV_BASE_DIRRTL`) la barra di scorrimento verticale verrà posizionata a sinistra. Si noti che la proprietà di stile `base_dir` viene ereditata. Pertanto, può essere impostato direttamente sulla parte `LV_PART_SCROLLBAR` di un Widget, o sulla parte `LV_PART_MAIN` del Widget, o su quella di uno qualsiasi dei suoi elementi padre, per far sì che una barra di scorrimento erediti la direzione di base.

`pad_left/right/top/bottom` sets the spacing around the scrollbars, `width` sets the scrollbar's width and `length` sets the scrollbar's length: If `length` is not set or left at 0 the scrollbar's length will be set automatically according to the length of the content.

```
static lv_style_t style_scrollbar;
lv_style_init(&style_scrollbar);
lv_style_set_pad_left(&style_scrollbar, 2);
lv_style_set_pad_right(&style_scrollbar, 2);
lv_style_set_pad_top(&style_scrollbar, 2);
lv_style_set_pad_bottom(&style_scrollbar, 2);
lv_style_set_width(&style_scrollbar, 10);
lv_style_set_length(&style_scrollbar, 50);
...
lv_obj_add_style(widget, &style_scrollbar, LV_PART_SCROLLBAR);
```

La lunghezza minima della barra di scorrimento è fissata a 10, mentre la lunghezza massima è limitata dall'altezza o dalla larghezza del widget, a seconda che la barra di scorrimento sia verticale o orizzontale. Qualsiasi valore di lunghezza impostato al di fuori di questi limiti comporterà automaticamente una lunghezza fissata a uno dei due limiti.

## Eventi dello Scrolling

I seguenti eventi vengono generati come parte dello scorrimento:

- `LV_EVENT_SCROLL_BEGIN`: Segnala l'inizio dello scrolling. Il parametro `event` è `NULL` o `lv_anim_t` \* con un descrittore di animazione di scorrimento che può essere modificato se necessario.
- `LV_EVENT_SCROLL_END`: Segnala la fine dello scrolling.
- `LV_EVENT_SCROLL`: Segnala la modifica della posizione di scorrimento; attivato a ogni cambio di posizione.

### 5.7.2 Funzionalità dello Scrolling

Oltre alla gestione dello scorrimento "normale", ci sono molte funzionalità aggiuntive interessanti e utili.

#### Scollable

È possibile rendere un Widget non-scrollabile con `lv_obj_remove_flag(widget, LV_OBJ_FLAG_SCROLLABLE)`.

I Widget non-scrollabile possono comunque propagare lo scorrimento (chain [catena]) ai widget padre.

La direzione dello scorrimento può essere controllata da `lv_obj_set_scroll_dir(widget, LV_DIR_...)`.

Per la direzione è possibile utilizzare i seguenti valori:

- `LV_DIR_TOP`: scorrimento solo verso l'alto
- `LV_DIR_LEFT`: scorrimento solo verso sinistra
- `LV_DIR_BOTTOM`: scorrimento solo verso il basso
- `LV_DIR_RIGHT`: scorrimento solo verso destra

- *LV\_DIR\_HOR*: scorimento solo orizzontalmente
- *LV\_DIR\_VER*: scorimento solo verticalmente
- *LV\_DIR\_ALL*: scorimento in qualsiasi direzione

Sono possibili anche valori in OR. Es. *LV\_DIR\_TOP* | *LV\_DIR\_LEFT*.

### Concatenamento dello Scroll

Se un widget non può essere ulteriormente scorrevole (ad esempio, perché il suo contenuto ha raggiunto la posizione più bassa), lo scorimento aggiuntivo viene propagato al suo elemento padre. Se il genitore può essere scorrevole in quella direzione, verrà invece "scrollato". Continua a propagarsi lungo la gerarchia del genitore del Widget fino a *Screen*.

La propagazione durante lo scorimento è chiamata "scroll chaining" e può essere abilitata/disabilitata con il flag *LV\_OBJ\_FLAG\_SCROLL\_CHAIN\_HOR/VER*. Se il chaining è disabilitato, la propagazione si interrompe sul Widget e il genitore/i non verrà fatto scorrere.

### Scorimento dinamico

Quando l'utente scorre un widget e lo rilascia, LVGL può emulare lo slancio inerziale per lo scorrimento. È come se il widget fosse stato "lanciato" e lo scorimento rallenta gradualmente.

Lo slancio dinamico può essere abilitato/disabilitato con il flag *LV\_OBJ\_FLAG\_SCROLL\_MOMENTUM*.

### Scorimento elastico

Normalmente non è possibile scorrere un Widget oltre le estremità del suo contenuto. Ovvero, il lato superiore del contenuto non può trovarsi al di sotto del lato superiore del Widget e viceversa per il lato inferiore.

Tuttavia, con *LV\_OBJ\_FLAG\_SCROLL\_ELASTIC* viene aggiunto un effetto particolare quando l'utente "scorre eccessivamente" il contenuto. Lo scorimento rallenta e il contenuto può essere fatto scorrere all'interno del Widget. Quando il Widget viene rilasciato, il contenuto che scorre al suo interno viene riportato alla posizione valida più vicina.

### Aggancio

I figli di un Widget possono essere agganciati secondo regole specifiche al termine dello scorrimento. I figli possono essere resi agganciabili individualmente con il flag *LV\_OBJ\_FLAG\_SNAPPABLE*.

Un Widget può allineare i figli agganciati in quattro modi:

- *LV\_SCROLL\_SNAP\_NONE*: Lo "Snap" [aggancio] è disabilitato. (il default)
- *LV\_SCROLL\_SNAP\_START*: Allinea gli elementi figlio a sinistra/in alto di un Widget scorrevole
- *LV\_SCROLL\_SNAP\_END*: Allinea gli elementi figlio a destra/in basso di un Widget scorrevole
- *LV\_SCROLL\_SNAP\_CENTER*: Allinea gli elementi figlio al centro di un Widget scorrevole

L'allineamento snap viene impostato con *lv\_obj\_set\_scroll\_snap\_x(widget, LV\_SCROLL\_SNAP\_...)* e *lv\_obj\_set\_scroll\_snap\_y(widget, LV\_SCROLL\_SNAP\_...)*.

Ecco cosa succede dietro le quinte:

1. l'utente scorre e rilascia un widget;
2. LVGL calcola dove terminerebbe lo scorrimento considerando la quantità di moto dello scorrimento;
3. LVGL trova il punto di scorrimento più vicino;
4. LVGL scorre fino al punto di aggancio con un'animazione.

## Scroll one

La funzione "scroll one" [scorrimento di uno] indica a LVGL di consentire lo scorrimento di un solo elemento figlio agganciabile alla volta. Ciò richiede di rendere gli elementi figlio agganciabili e di impostare l'allineamento dello snap dello scorrimento su un valore diverso da `LV_SCROLL_SNAP_NONE`.

Questa funzionalità può essere abilitata tramite il flag `LV_OBJ_FLAG_SCROLL_ONE`.

## Scorrimento con focus

Si immagini che ci siano molti Widget in un gruppo che si trovano su un Widget scorrevole. Premendo il pulsante "Tab" si sposta il focus sul Widget successivo, ma potrebbe trovarsi al di fuori dell'area visibile del Widget scorrevole. Se la funzione "scroll on focus" è abilitata, LVGL scorrerà automaticamente i Widget per visualizzare il Widget figlio con focus. Lo scorrimento avviene in modo ricorsivo, quindi anche i Widget scorrevoli annidati vengono gestiti correttamente. Il Widget verrà visualizzato anche se si trova su una pagina diversa di una visualizzazione a schede.

### 5.7.3 Scorrimento Programmatico

Le seguenti funzioni API consentono lo scorrimento programmatico dei Widget:

- `lv_obj_scroll_by(widget, x, y, LV_ANIM_ON/OFF)` scorre in base ai valori x e y
- `lv_obj_scroll_to(widget, x, y, LV_ANIM_ON/OFF)` scorre per portare la coordinata specificata nell'angolo in alto a sinistra
- `lv_obj_scroll_to_x(widget, x, LV_ANIM_ON/OFF)` scorre per portare la coordinata specificata sul lato sinistro
- `lv_obj_scroll_to_y(widget, y, LV_ANIM_ON/OFF)` scorre per portare la coordinata specificata sul lato superiore

Di tanto in tanto potrebbe essere necessario recuperare la *posizione di scorrimento* di un widget scorrevole, per ripristinarla in seguito o per visualizzarla dinamicamente alcuni elementi in base alla loro posizione di scorrimento corrente. Ecco un esempio per illustrare come combinare l'evento di scorrimento e memorizzare la posizione di scorrimento superiore.

```
static int scroll_value = 0;

static void store_scroll_top_value_event_cb(lv_event_t* e) {
    lv_obj_t * scr = lv_event_get_target(e);
    scroll_value = lv_obj_get_scroll_top(scr);
    printf("%d pixels are scrolled above top edge of display.\n", scroll_value);
}

lv_obj_t * scr = lv_obj_create(NULL);
lv_obj_add_event_cb(scr, store_scroll_top_value_event_cb, LV_EVENT_SCROLL, NULL);
```

Le coordinate di scorrimento possono essere recuperate da assi diversi con queste funzioni:

- `lv_obj_get_scroll_x(widget)` Pixel che hanno superato il bordo sinistro della finestra di visualizzazione del Widget.
- `lv_obj_get_scroll_y(widget)` Pixel che hanno superato la parte superiore della finestra di visualizzazione del Widget.
- `lv_obj_get_scroll_top(widget)` Identico a `lv_obj_get_scroll_y(widget)`
- `lv_obj_get_scroll_bottom(widget)` Pixel che hanno superato la parte inferiore della finestra di visualizzazione del Widget.
- `lv_obj_get_scroll_left(widget)` Identico a `lv_obj_get_scroll_x(widget)`.
- `lv_obj_get_scroll_right(widget)` Pixel scorsi oltre il bordo destro della finestra di visualizzazione del Widget.

L'impostazione della posizione di scorrimento può essere effettuata con queste funzioni:

- `lv_obj_scroll_by(widget, dx, dy, anim_enable)` Scorrimento di un numero specificato di pixel.
- `lv_obj_scroll_by_bounded(widget, dx, dy, animation_enable)` Scorrimento di un numero specificato di pixel.

- `lv_obj_scroll_to(widget, x, y, animation_enable)` Scorrimento fino alla coordinata specificata sul Widget.
- `lv_obj_scroll_to_x(widget, x, animation_enable)` Scorri fino alla coordinata X sul Widget.
- `lv_obj_scroll_to_y(widget, y, animation_enable)` corri fino alla coordinata Y sul Widget.
- `lv_obj_scroll_to_view(widget, animation_enable)` Scorri il Widget padre di obj finché obj non diventa visibile.
- `lv_obj_scroll_to_view_recursive(widget, animation_enable)` Scorre ricorsivamente i widget padre di obj finché obj non diventa visibile.

## 5.7.4 Dimensione Automatica

La "Self size" [dimensione automatica] è una proprietà di un Widget. Normalmente, l'utente non dovrebbe utilizzare questo parametro, ma se si crea un widget personalizzato potrebbe essere utile.

In breve, il self size stabilisce la dimensione del contenuto di un Widget. Per capirlo meglio, prendiamo l'esempio di una tabella. Supponiamo che abbia 10 righe, ciascuna con un'altezza di 50 px. Quindi l'altezza totale del contenuto è di 500 px. In altre parole, la "self height" è di 500 px. Se l'utente imposta solo un'altezza di 200 px per la tabella, LVGL vedrà che la dimensione automatica è maggiore e renderà la tabella scorrevole.

Ciò significa che non solo i figli possono rendere scorrevole un Widget, ma anche una dimensione automatica maggiore.

LVGL utilizza l'evento `LV_EVENT_GET_SELF_SIZE` per ottenere la dimensione automatica di un Widget. Ecco un esempio per vedere come gestire l'evento:

```
if(event_code == LV_EVENT_GET_SELF_SIZE) {
    lv_point_t * p = lv_event_get_param(e);

    /* If x or y < 0 then it doesn't need to be calculated now. */
    if(p->x >= 0) {
        p->x = 200; /* Set or calculate self width. */
    }

    if(p->y >= 0) {
        p->y = 50; /* Set or calculate self height. */
    }
}
```

## 5.7.5 Esempi

### Scrolling nidificato

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

static lv_obj_t * panel;
static lv_obj_t * save_button;
static lv_obj_t * restore_button;
static int     saved_scroll_x;
static int     saved_scroll_y;

static void scroll_update_cb(lv_event_t * e);
static void button_event_cb(lv_event_t * e);

static void scroll_update_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    LV_LOG("scroll info: x:%3" LV_PRId32", y:%3" LV_PRId32", top:%3" LV_PRId32", "
          "bottom:%3" LV_PRId32", left:%3" LV_PRId32", right:%3" LV_PRId32"\n",
          lv_obj_get_scroll_x(panel),
          lv_obj_get_scroll_y(panel),
          lv_obj_get_scroll_top(panel),
          lv_obj_get_scroll_bottom(panel),
          lv_obj_get_scroll_left(panel),
          lv_obj_get_scroll_right(panel));
}

static void button_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    if(obj == save_button) {
        saved_scroll_x = lv_obj_get_scroll_x(panel);
        saved_scroll_y = lv_obj_get_scroll_y(panel);
    }
    else {
        lv_obj_scroll_to(panel, saved_scroll_x, saved_scroll_y, LV_ANIM_ON);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    }

    /**
     * Demonstrate how scrolling appears automatically
     */
void lv_example_scroll_1(void)
{
    /*Create an object with the new style*/
    lv_obj_t * scr;
    scr = lv_screen_active();
    panel = lv_obj_create(scr);
    lv_obj_set_size(panel, 200, 200);
    lv_obj_align(panel, LV_ALIGN_CENTER, 44, 0);

    lv_obj_t * child;
    lv_obj_t * label;

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 0, 0);
    lv_obj_set_size(child, 70, 70);
    label = lv_label_create(child);
    lv_label_set_text(label, "Zero");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 160, 80);
    lv_obj_set_size(child, 80, 80);

    lv_obj_t * child2 = lv_button_create(child);
    lv_obj_set_size(child2, 100, 50);

    label = lv_label_create(child2);
    lv_label_set_text(label, "Right");
    lv_obj_center(label);

    child = lv_obj_create(panel);
    lv_obj_set_pos(child, 40, 160);
    lv_obj_set_size(child, 100, 70);
    label = lv_label_create(child);
    lv_label_set_text(label, "Bottom");
    lv_obj_center(label);

    /* When LV_OBJ_FLAG_SCROLL_ELASTIC is cleared, scrolling does not go past edge boundaries. */
    /* lv_obj_clear_flag(panel, LV_OBJ_FLAG_SCROLL_ELASTIC); */

    /* Call `scroll_update_cb` while panel is being scrolled. */
    lv_obj_add_event_cb(panel, scroll_update_cb, LV_EVENT_SCROLL, NULL);

    /* Set up buttons that save and restore scroll position. */
    save_button = lv_button_create(scr);
    restore_button = lv_button_create(scr);
    lv_obj_t * lbl;
    lbl = lv_label_create(save_button);
    lv_label_set_text_static(lbl, "Save");
    lbl = lv_label_create(restore_button);
    lv_label_set_text_static(lbl, "Restore");
    lv_obj_align_to(save_button, panel, LV_ALIGN_OUT_LEFT_MID, -10, -20);
    lv_obj_align_to(restore_button, panel, LV_ALIGN_OUT_LEFT_MID, -10, 20);
    lv_obj_add_event_cb(save_button, button_event_cb, LV_EVENT_CLICKED, NULL);
    lv_obj_add_event_cb(restore_button, button_event_cb, LV_EVENT_CLICKED, NULL);
}

#endif

```

## Aggancio

```

#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES && LV_USE_FLEX

static void sw_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * sw = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_t * list = (lv_obj_t *) lv_event_get_user_data(e);

        if(lv_obj_has_state(list, LV_STATE_CHECKED)) lv_obj_add_flag(list, LV_OBJ_FLAG_SCROLL_ONE);
        else lv_obj_remove_flag(list, LV_OBJ_FLAG_SCROLL_ONE);
    }
}

/**
 * Show an example to scroll snap
 */
void lv_example_scroll_2(void)
{
    lv_obj_t * panel = lv_obj_create(lv_screen_active());
    lv_obj_set_size(panel, 280, 120);
    lv_obj_set_scroll_snap_x(panel, LV_SCROLL_SNAP_CENTER);
    lv_obj_set_flex_flow(panel, LV_FLEX_FLOW_ROW);
    lv_obj_align(panel, LV_ALIGN_CENTER, 0, 20);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * btn = lv_button_create(panel);
        lv_obj_set_size(btn, 150, lv_pct(100));

        lv_obj_t * label = lv_label_create(btn);
        if(i == 3) {
            lv_label_set_text_fmt(label, "Panel %" LV_PRIu32"\nno snap", i);
            lv_obj_remove_flag(btn, LV_OBJ_FLAG_SNAPPABLE);
        }
        else {
            lv_label_set_text_fmt(label, "Panel %" LV_PRIu32, i);
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_obj_center(label);
}
lv_obj_update_snap(panel, LV_ANIM_ON);

#if LV_USE_SWITCH
/*Switch between "One scroll" and "Normal scroll" mode*/
lv_obj_t * sw = lv_switch_create(lv_screen_active());
lv_obj_align(sw, LV_ALIGN_TOP_RIGHT, -20, 10);
lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_ALL, panel);
lv_obj_t * label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "One scroll");
lv_obj_align_to(label, sw, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
#endif
}
#endif
}

```

## Pulsante mobile

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST

static uint32_t btn_cnt = 1;

static void float_button_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * float_btn = lv_event_get_target_obj(e);

    if(code == LV_EVENT_CLICKED) {
        lv_obj_t * list = (lv_obj_t *) lv_event_get_user_data(e);
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_obj_t * list_btn = lv_list_add_button(list, LV_SYMBOL_AUDIO, buf);
        btn_cnt++;
    }

    lv_obj_move_foreground(float_btn);

    lv_obj_scroll_to_view(list_btn, LV_ANIM_ON);
}
}

/***
 * Create a list with a floating button
 */
void lv_example_scroll_3(void)
{
    lv_obj_t * list = lv_list_create(lv_screen_active());
    lv_obj_set_size(list, 280, 220);
    lv_obj_center(list);

    for(btn_cnt = 1; btn_cnt <= 2; btn_cnt++) {
        char buf[32];
        lv_snprintf(buf, sizeof(buf), "Track %d", (int)btn_cnt);
        lv_list_add_button(list, LV_SYMBOL_AUDIO, buf);
    }

    lv_obj_t * float_btn = lv_button_create(list);
    lv_obj_set_size(float_btn, 50, 50);
    lv_obj_add_flag(float_btn, LV_OBJ_FLAG_FLOATING);
    lv_obj_align(float_btn, LV_ALIGN_BOTTOM_RIGHT, 0, -lv_obj_get_style_pad_right(list, LV_PART_MAIN));
    lv_obj_add_event_cb(float_btn, float_button_event_cb, LV_EVENT_ALL, list);
    lv_obj_set_style_radius(float_btn, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_image_src(float_btn, LV_SYMBOL_PLUS, 0);
    lv_obj_set_style_text_font(float_btn, lv_theme_get_font_large(float_btn), 0);
}
#endif

```

## Stile delle barre di scorrimento

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LIST

/***
 * Styling the scrollbars
 */
void lv_example_scroll_4(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_set_size(obj, 200, 100);
    lv_obj_center(obj);

    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text(label,
        "Lorem ipsum dolor sit amet, consectetur adipiscing elit.\n"
        "Etiam dictum, tortor vestibulum lacinia laoreet, mi neque consectetur neque, vel mattis odio dolor egestas ligula. \n"
        "Sed vestibulum sapien nulla, id convallis ex porttitor nec. \n"
        "Duis et massa eu libero accumsan faucibus a in arcu. \n"
        "Ut pulvinar odio lorem, vel tempus turpis condimentum quis. Nam consectetur condimentum sem in auctor. \n"
        "Sed nisl augue, venenatis in blandit et, gravida ac tortor. \n"
        "Etiam dapibus elementum suscipit. \n"
        "Proin mollis sollicitudin convallis. \n"
        "Integer dapibus tempus arcu nec viverra. \n"
        "Donec molestie nulla enim, eu interdum velit placerat quis. \n"
        "Donec id efficitur risus, at molestie turpis. \n"
        "Suspendisse vestibulum consectetur nunc ut commodo. \n"
        "Fusce molestie rhoncus nisi sit amet tincidunt. \n"
        "Suspendisse a nunc ut magna ornare volutpat.");
}

/*Remove the style of scrollbar to have clean start/
lv_obj_remove_style(obj, NULL, LV_PART_SCROLLBAR | LV_STATE_ANY);

/*Create a transition the animate the some properties on state change*/

```

(continues on next page)

(continua dalla pagina precedente)

```

static const lv_style_prop_t props[] = {LV_STYLE_BG_OPA, LV_STYLE_WIDTH, 0};
static lv_style_transition_dsc_t trans;
lv_style_transition_dsc_init(&trans, props, lv_anim_path_linear, 200, 0, NULL);

/*Create a style for the scrollbars*/
static lv_style_t style;
lv_style_init(&style);
lv_style_set_width(&style, 4);           /*Width of the scrollbar*/
lv_style_set_length(&style, 20);         /*Length of the scrollbar*/
lv_style_set_pad_right(&style, 5);       /*Space from the parallel side*/
lv_style_set_pad_top(&style, 5);         /*Space from the perpendicular side*/

lv_style_set_radius(&style, 2);
lv_style_set_bg_opa(&style, LV_OPA_70);
lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
lv_style_set_border_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 3));
lv_style_set_border_width(&style, 2);
lv_style_set_shadow_width(&style, 8);
lv_style_set_shadow_spread(&style, 2);
lv_style_set_shadow_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 1));

lv_style_set_transition(&style, &trans);

/*Make the scrollbars wider and use 100% opacity when scrolled*/
static lv_style_t style_scrolled;
lv_style_init(&style_scrolled);
lv_style_set_width(&style_scrolled, 8);
lv_style_set_bg_opa(&style_scrolled, LV_OPA_COVER);

lv_obj_add_style(obj, &style, LV_PART_SCROLLBAR);
lv_obj_add_style(obj, &style_scrolled, LV_PART_SCROLLBAR | LV_STATE_SCROLLED);

}

#endif

```

## Scorrimento da destra a sinistra

## Traslazione durante lo scorrimento

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static void scroll_event_cb(lv_event_t * e)
{
    lv_obj_t * cont = lv_event_get_target_obj(e);

    lv_area_t cont_a;
    lv_obj_get_coords(cont, &cont_a);
    int32_t cont_y_center = cont_a.y1 + lv_area_get_height(&cont_a) / 2;

    int32_t r = lv_obj_get_height(cont) * 7 / 10;
    int32_t i;
    int32_t child_cnt = (int32_t)lv_obj_get_child_count(cont);
    for(i = 0; i < child_cnt; i++) {
        lv_obj_t * child = lv_obj_get_child(cont, i);
        lv_area_t child_a;
        lv_obj_get_coords(child, &child_a);

        int32_t child_y_center = child_a.y1 + lv_area_get_height(&child_a) / 2;

        int32_t diff_y = child_y_center - cont_y_center;
        diff_y = LV_ABS(diff_y);

        /*Get the x of diff_y on a circle.*/
        int32_t x;
        /*If diff_y is out of the circle use the last point of the circle (the radius)*/
        if(diff_y >= r) {
            x = r;
        }
        else {
            /*Use Pythagoras theorem to get x from radius and y*/
            uint32_t x_sqr = r * r - diff_y * diff_y;
            lv_sqrt_res_t res;
            lv_sqrt(x_sqr, &res, 0x8000);    /*Use lvgl's built in sqrt root function*/
            x = r - res.i;
        }
    }

    /*Translate the item by the calculated X coordinate*/
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_style_translate_x(child, x, 0);

/*Use some opacity with larger translations*/
lv_opa_t opa = (lv_opa_t)lv_map(x, 0, r, LV_OPA_TRANS, LV_OPA_COVER);
lv_obj_set_style_opa(child, LV_OPA_COVER - opa, 0);
}

/***
 * Translate the object as they scroll
 */
void lv_example_scroll_6(void)
{
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, 200, 200);
    lv_obj_center(cont);
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN);
    lv_obj_add_event_cb(cont, scroll_event_cb, LV_EVENT_SCROLL, NULL);
    lv_obj_set_style_radius(cont, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(cont, true, 0);
    lv_obj_set_scroll_dir(cont, LV_DIR_VERT);
    lv_obj_set_scroll_snap_y(cont, LV_SCROLL_SNAP_CENTER);
    lv_obj_set_scrollbar_mode(cont, LV_SCROLLBAR_MODE_OFF);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        lv_obj_t * btn = lv_button_create(cont);
        lv_obj_set_width(btn, lv_pct(100));

        lv_obj_t * label = lv_label_create(btn);
        lv_label_set_text_fmt(label, "Button %" LV_PRIu32, i);
    }

    /*Update the buttons position manually for first*/
    lv_obj_send_event(cont, LV_EVENT_SCROLL, NULL);

    /*Be sure the fist button is in the middle*/
    lv_obj_scroll_to_view(lv_obj_get_child(cont, 0), LV_ANIM_OFF);
}
#endif

```

## Scrolling infinito

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

static lv_obj_t * high_label;
static lv_obj_t * low_label;
static int32_t top_num;
static int32_t bottom_num;
static bool update_scroll_running = false;

static lv_obj_t * load_item(lv_obj_t * parent, int32_t num)
{
    lv_obj_t * obj = (lv_obj_t *) lv_obj_create(parent);
    lv_obj_set_size(obj, LV_PCT(100), LV_SIZE_CONTENT);
    lv_obj_t * label = lv_label_create(obj);
    lv_label_set_text_fmt(label, "%" PRId32, num);
    return obj;
}

static void update_scroll(lv_obj_t * obj)
{
    /* do not re-enter this function when `lv_obj_scroll_by` triggers this callback again. */
    if(update_scroll_running) return;
    update_scroll_running = true;

    int32_t top_num_original = top_num;
    int32_t bottom_num_original = bottom_num;

    /* load items we're getting close to */
    while(bottom_num > -30 && lv_obj_get_scroll_bottom(obj) < 200) {
        bottom_num -= 1;
        load_item(obj, bottom_num);
        lv_obj_update_layout(obj);
        LV_LOG_USER("loaded bottom num: %" PRId32, bottom_num);
    }
    while(top_num < 30 && lv_obj_get_scroll_top(obj) < 200) {
        top_num += 1;
        int32_t bottom_before = lv_obj_get_scroll_bottom(obj);
        lv_obj_t * new_item = load_item(obj, top_num);
        lv_obj_move_to_index(new_item, 0);
        lv_obj_update_layout(obj);
        int32_t bottom_after = lv_obj_get_scroll_bottom(obj);
        lv_obj_scroll_by(obj, 0, bottom_before - bottom_after, LV_ANIM_OFF);
        LV_LOG_USER("loaded top num: %" PRId32, top_num);
    }

    /* delete far-away items */
    while(lv_obj_get_scroll_bottom(obj) > 600) {
        bottom_num += 1;
        lv_obj_t * child = lv_obj_get_child(obj, -1);
        lv_obj_delete(child);
        lv_obj_update_layout(obj);
        LV_LOG_USER("deleted bottom num: %" PRId32, bottom_num);
    }
    while(lv_obj_get_scroll_top(obj) > 600) {
        top_num -= 1;
        int32_t bottom_before = lv_obj_get_scroll_bottom(obj);
        lv_obj_t * child = lv_obj_get_child(obj, 0);
        lv_obj_delete(child);
        lv_obj_update_layout(obj);
        int32_t bottom_after = lv_obj_get_scroll_bottom(obj);
        lv_obj_scroll_by(obj, 0, bottom_before - bottom_after, LV_ANIM_OFF);
        LV_LOG_USER("deleted top num: %" PRId32, top_num);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    }

    if(top_num != top_num_original) {
        lv_label_set_text_fmt(high_label, "current largest\nloaded value:\n%" PRId32, top_num);
    }
    if(bottom_num != bottom_num_original) {
        lv_label_set_text_fmt(low_label, "current smallest\nloaded value:\n%" PRId32, bottom_num);
    }

    update_scroll_running = false;
}

static void scroll_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    update_scroll(obj);
}

static void checkbox_cb(lv_event_t * e)
{
    lv_obj_t * checkbox = lv_event_get_target_obj(e);
    lv_obj_t * obj = (lv_obj_t *) lv_event_get_user_data(e);
    bool checked = lv_obj_has_state(checkbox, LV_STATE_CHECKED);
    lv_obj_set_style_opa(obj, checked ? LV_OPA_COVER : LV_OPA_TRANS, LV_PART_SCROLLBAR);
}

/**
 * Dynamically load widgets while scrolling
 */
void lv_example_scroll_7(void)
{
    lv_obj_t * scr = lv_screen_active();
    lv_obj_t * obj = lv_obj_create(scr);
    lv_obj_set_size(obj, 160, 220);
    lv_obj_align(obj, LV_ALIGN_RIGHT_MID, -10, 0);
    lv_obj_set_flex_flow(obj, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_style_opa(obj, LV_OPA_TRANS, LV_PART_SCROLLBAR);

    high_label = lv_label_create(scr);
    lv_label_set_text_static(high_label, "current largest\nloaded value:");
    lv_obj_align(high_label, LV_ALIGN_TOP_LEFT, 10, 10);

    lv_obj_t * checkbox = lv_checkbox_create(scr);
    lv_checkbox_set_text_static(checkbox, "show\nscrollbar");
    lv_obj_align(checkbox, LV_ALIGN_LEFT_MID, 10, 0);
    lv_obj_add_event_cb(checkbox, checkbox_cb, LV_EVENT_VALUE_CHANGED, obj);

    low_label = lv_label_create(scr);
    lv_label_set_text_static(low_label, "current smallest\nloaded value:");
    lv_obj_align(low_label, LV_ALIGN_BOTTOM_LEFT, 10, -10);

    load_item(obj, 3);
    /* These counters hold the highest/lowest number currently loaded. */
    top_num = 3;
    bottom_num = 3;

    lv_obj_update_layout(obj);
    update_scroll(obj);
    lv_obj_add_event_cb(obj, scroll_cb, LV_EVENT_SCROLL, NULL);
}

#endif

```

## Scrolling circolare

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_FLEX

/*
 * Circular list implementation based on:
 * Reference: https://blog.csdn.net/TQW4321/article/details/145434819
 */

/* Applies to both width (for row items) and height (for column items) */
#define ITEM_SIZE     80

/* Function to calculate the total content width of the container */
static int32_t get_content_width(lv_obj_t * cont)
{
    int32_t w = 0;
    int32_t pad_column = lv_obj_get_style_pad_column(cont, LV_PART_MAIN);
    int32_t total = (int32_t)lv_obj_get_child_count(cont);

    for(int32_t i = 0; i < total; i++) {
        w += lv_obj_get_width(lv_obj_get_child(cont, i));
        if(i < total - 1) w += pad_column;
    }
    return w + lv_obj_get_style_pad_left(cont, LV_PART_MAIN)
        + lv_obj_get_style_pad_right(cont, LV_PART_MAIN);
}

/* Scroll event callback for row layout */
static void cont_row_scroll_event_cb(lv_event_t * e)
{
    static bool is_adjusting = false;
    lv_obj_t * cont = lv_event_get_current_target_obj(e);

    if(!is_adjusting) {
        is_adjusting = true;
        int32_t scroll_x = lv_obj_get_scroll_x(cont);
        int32_t cont_w = lv_obj_get_width(cont);
        int32_t content_w = get_content_width(cont);

        /* Use ITEM_SIZE as horizontal item width */
        const int32_t item_width = ITEM_SIZE;

        if(scroll_x <= 0) {

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * last_child = lv_obj_get_child(cont, (int32_t)(lv_obj_get_child_count(cont) - 1));
lv_obj_move_to_index(last_child, 0);
lv_obj_scroll_to_x(cont, scroll_x + item_width, LV_ANIM_OFF);
}
else if(scroll_x > content_w - cont_w) {
    lv_obj_t * first_child = lv_obj_get_child(cont, 0);
    lv_obj_move_to_index(first_child, (int32_t)(lv_obj_get_child_count(cont) - 1));
    lv_obj_scroll_to_x(cont, scroll_x - item_width, LV_ANIM_OFF);
}
is_adjusting = false;
}

/* Function to calculate the total content height of the container */
static int32_t get_content_height(lv_obj_t * cont)
{
    int32_t h = 0;
    int32_t pad_row = lv_obj_get_style_pad_row(cont, LV_PART_MAIN);
    int32_t total = (int32_t)lv_obj_get_child_count(cont);

    for(int i = 0; i < total; i++) {
        h += lv_obj_get_height(lv_obj_get_child(cont, i));
        if(i < total - 1) h += pad_row;
    }
    return h + lv_obj_get_style_pad_top(cont, LV_PART_MAIN)
        + lv_obj_get_style_pad_bottom(cont, LV_PART_MAIN);
}

/* Scroll event callback for column layout */
static void cont_col_scroll_event_cb(lv_event_t * e)
{
    static bool is_adjusting = false;
    lv_obj_t * cont = lv_event_get_current_target_obj(e);

    if(!is_adjusting) {
        is_adjusting = true;
        int32_t scroll_y = lv_obj_get_scroll_y(cont);
        int32_t cont_h = lv_obj_get_height(cont);
        int32_t content_h = get_content_height(cont);

        /* Use ITEM_SIZE as vertical item height */
        const int32_t item_height = ITEM_SIZE;

        if(scroll_y <= 0) {
            lv_obj_t * last_child = lv_obj_get_child(cont, (int32_t)(lv_obj_get_child_count(cont) - 1));
            lv_obj_move_to_index(last_child, 0);
            lv_obj_scroll_to_y(cont, scroll_y + item_height, LV_ANIM_OFF);
        }
        else if(scroll_y > content_h - cont_h) {
            lv_obj_t * first_child = lv_obj_get_child(cont, 0);
            lv_obj_move_to_index(first_child, (int32_t)(lv_obj_get_child_count(cont) - 1));
            lv_obj_scroll_to_y(cont, scroll_y - item_height, LV_ANIM_OFF);
        }
        is_adjusting = false;
    }
}

void lv_example_scroll_8(void)
{
    /* Create a scroll container with ROW flex direction */
    lv_obj_t * cont_row = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_row, 300, 75);
    lv_obj_align(cont_row, LV_ALIGN_TOP_MID, 0, 5);
    lv_obj_set_flex_flow(cont_row, LV_FLEX_FLOW_ROW);
    lv_obj_add_event_cb(cont_row, cont_row_scroll_event_cb, LV_EVENT_SCROLL, NULL);
    /* Hide scrollbar visuals */
    lv_obj_set_scrollbar_mode(cont_row, LV_SCROLLBAR_MODE_OFF);

    /* Create a scroll container with COLUMN flex direction */
    lv_obj_t * cont_col = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont_col, 200, 150);
    lv_obj_align_to(cont_col, cont_row, LV_ALIGN_OUT_BOTTOM_MID, 0, 5);
    lv_obj_set_flex_flow(cont_col, LV_FLEX_FLOW_COLUMN);
    lv_obj_add_event_cb(cont_col, cont_col_scroll_event_cb, LV_EVENT_SCROLL, NULL);
    /* Hide scrollbar visuals */
    lv_obj_set_scrollbar_mode(cont_col, LV_SCROLLBAR_MODE_OFF);

    /* If the number of items is less than 3, scrolling may not be needed or may cause unexpected behavior. */
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * obj;
        lv_obj_t * label;

        /*Add items to the row*/
        obj = lv_button_create(cont_row);
        lv_obj_set_size(obj, ITEM_SIZE, LV_PCT(100));

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item %" LV_PRIu32, i + 1);
        lv_obj_center(label);

        /*Add items to the column*/
        obj = lv_button_create(cont_col);
        lv_obj_set_size(obj, LV_PCT(100), ITEM_SIZE);

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "Item %" LV_PRIu32, i + 1);
        lv_obj_center(label);
    }
}

#endif

```

## 5.7.6 API

# CAPITOLO 6

---

## Tutti i Widget

---

### 6.1 Widget Base (lv\_obj)

Il più fondamentale di tutti i Widget è il Widget Base, su cui si basano tutti gli altri. Da una prospettiva Object-Oriented, si può pensare al Widget Base come alla classe Widget da cui ereditano tutti gli altri Widget.

In questo modo, tutti i Widget portano con sé *le funzionalità del Widget Base*. Pertanto, le funzioni e le funzionalità del Widget Base possono essere utilizzate anche con altri widget. Ad esempio `lv_obj_set_width(slider, 100)`.

Il Widget Base può essere utilizzato direttamente come un semplice widget. Sebbene sia un semplice rettangolo, presenta numerose funzionalità condivise con tutti i Widget, descritte in dettaglio in *Funzionalità Comuni dei Widget*. In termini HTML, lo si immagini come `<div>`.

### 6.2 3D Texture (lv\_3dtexture)

#### 6.2.1 Panoramica

I widget texture 3D vengono utilizzati per incorporare la primitiva "texture" di una libreria grafica 3D esterna in un'interfaccia utente LVGL.

#### 6.2.2 Parti e Stili

La dimensione deve essere impostata sulla dimensione effettiva della primitiva texture. Anche l'opacità può essere modificata.

#### 6.2.3 Utilizzo

Richiede l'abilitazione di un'unità di disegno in grado di disegnare il tipo di attività `LV_DRAW_TASK_TYPE_3D`.

OpenGL è il primo back-end grafico 3D supportato. È necessario abilitare quanto segue.

- `LV_USE_3DTEXTURE`
- `LV_USE_OPENGLS`
- `LV_USE_DRAW_OPENGLS`

Consultare *la documentazione del driver OpenGL di LVGL* per creare una finestra e una texture di visualizzazione.

Il repository `lv_example_3dtexture` <[https://github.com/lvgl/lv\\_example\\_3dtexture](https://github.com/lvgl/lv_example_3dtexture)> è una dimostrazione di come utilizzare il widget `3dtexture` per caricare modelli glTF con OpenGL e visualizzarli in LVGL.

```
lv_obj_t * tex = lv_3dtexture_create(parent);
/* Render something to the texture. You can replace it with your code. */
lv_3dtexture_id_t gltf_texture = render_gltf_model_to_opengl_texture(path, w, h, color);
lv_3dtexture_set_src(tex, gltf_texture);
lv_obj_set_size(tex, w, h);
lv_obj_set_style(tex, opa, 0);
```

Il tipo effettivo di `lv_3dtexture_id_t` dipende dal back-end grafico 3D attivo. Con OpenGL, il tipo è `unsigned int`.

## 6.2.4 Eventi

La Texture 3D non ha una gestione speciale degli eventi.

## 6.2.5 Esempio

Vedere il repository [lv\\_example\\_3dtexture](https://github.com/lvgl/lv_example_3dtexture) <[https://github.com/lvgl/lv\\_example\\_3dtexture](https://github.com/lvgl/lv_example_3dtexture)>.

## 6.2.6 API

`lv_3dtexture.h`

`lv_3dtexture_private.h`

# 6.3 Immagine di Animazione (lv\_animimg)

## 6.3.1 Panoramica

L'immagine di animazione è simile al normale Widget 'Image'. L'unica differenza è che invece di un'immagine sorgente, si imposta un array di più immagini sorgente che forniscono i "fotogrammi" in un'animazione.

È possibile specificare una durata e un numero di ripetizioni.

## 6.3.2 Parti e Stili

- `LV_PART_MAIN` Un rettangolo di background che utilizza *tipiche proprietà di stile del background* e l'immagine stessa che utilizza le proprietà di stile dell'immagine.

## 6.3.3 Utilizzo

### Sorgenti delle immagini

Per impostare le sorgenti delle immagini animate, utilizzare `lv_animimg_set_src(animimg, dsc[], num)`.

Per impostare la sorgente delle immagini per la riproduzione al contrario dell'immagine animata, si usa `lv_animimg_set_src_reverse(animimg, dsc[], num)`.

### Utilizzo dell'animazione interna

Per casi d'uso più avanzati, l'animazione utilizzata internamente dall'immagine può essere recuperata utilizzando `lv_animimg_get_anim(animimg)`. In questo modo, è possibile utilizzare le funzioni `Animation`, ad esempio, per sovrascrivere i valori dell'animazione utilizzando `lv_anim_set_values(anim, start, end)` o per impostare una callback sull'evento di completamento dell'animazione.

### 6.3.4 Eventi

Nessun evento speciale viene inviato dai Widget Animation-Image.

#### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

### 6.3.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Animation-Image.

#### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

### 6.3.6 Esempio

#### Immagine animata semplice

```
#include "../../lv_examples.h"
#if LV_USE_ANIMIMG && LV_BUILD_EXAMPLES
LV_IMAGE_DECLARE(animimg001);
LV_IMAGE_DECLARE(animimg002);
LV_IMAGE_DECLARE(animimg003);

static const lv_image_dsc_t * anim_imgs[3] = {
    &animimg001,
    & animimg002,
    & animimg003,
};

void lv_example_animating_1(void)
{
    lv_obj_t * animimg0 = lv_animimg_create(lv_screen_active());
    lv_obj_center(animimg0);
    lv_animimg_set_src(animimg0, (const void **) anim_imgs, 3);
    lv_animimg_set_duration(animimg0, 1000);
    lv_animimg_set_repeat_count(animimg0, LV_ANIM_REPEAT_INFINITE);
    lv_animimg_start(animimg0);
}
#endif
```

### 6.3.7 API

*lv\_animimage.h*

*lv\_animimage\_private.h*

## 6.4 Arco (lv\_arc)

### 6.4.1 Panoramica

L'Arco è costituito da un arco di sfondo e da un arco in primo piano. L'arco in primo piano (indicatore) può essere regolato tramite touch.

### 6.4.2 Parti e Stili

- **LV\_PART\_MAIN** Disegna uno sfondo utilizzando le tipiche proprietà di stile del background e un arco utilizzando le proprietà di stile dell'arco. Le dimensioni e la posizione dell'Arco rispetteranno le proprietà di stile *padding*.
- **LV\_PART\_INDICATOR** Disegna un altro arco utilizzando le proprietà di stile *Arc*. I suoi valori di padding vengono interpretati in relazione all'Arco di sfondo.

- *LV\_PART\_KNOB* Disegna una maniglia all'estremità dell'indicatore utilizzando tutte le proprietà di sfondo e i valori di padding. Con padding zero, la dimensione della manopola è uguale alla larghezza dell'indicatore. Un padding maggiore lo rende più grande, un padding minore lo rende più piccolo.

### 6.4.3 Utilizzo

#### Valore e intervallo

È possibile impostare un nuovo valore utilizzando `lv_arc_set_value(arc, new_value)`. Il valore viene interpretato in un "range" [intervallo] (valori minimo e massimo) che può essere modificato con `lv_arc_set_range(arc, min, max)`. L'intervallo di default è 0..100.

L'Arco indicatore viene disegnato sull'Arco della parte principale. Quindi, se il valore è impostato sul massimo, l'indicatore coprirà l'intero Arco "di sfondo". Per impostare l'angolo iniziale e finale dell'Arco di sfondo, utilizzare una di queste funzioni:

- `lv_arc_set_bg_start_angle(arc, angle)`
- `lv_arc_set_bg_end_angle(arc, angle)`
- `lv_arc_set_bg_angles(arc, start_angle, end_angle)`

Zero gradi si trova al centro a destra (ore 3) del Widget e i gradi aumentano in senso orario. I valori degli angoli devono essere compresi tra [0 e 360].

#### Rotation

È possibile aggiungere un offset alla posizione a 0 gradi con `lv_arc_set_rotation(arc, deg)`.

#### Modalità

L'Arco può essere una delle seguenti modalità:

- *LV\_ARC\_MODE\_NORMAL* L'arco indicatore viene disegnato in senso orario dal valore minimo al valore corrente.
- *LV\_ARC\_MODE\_REVERSE* L'Arco indicatore viene disegnato in senso antiorario dal valore massimo al valore corrente.
- *LV\_ARC\_MODE\_SYMMETRICAL* L'Arco indicatore viene disegnato dal punto centrale al valore corrente.

La modalità può essere impostata tramite `lv_arc_set_mode(arc, LV_ARC_MODE_...)` e non ha effetto finché l'angolo non viene impostato tramite `lv_arc_set_value()` o il valore dell'arco non viene modificato tramite l'input del puntatore (dito, mouse, ecc.).

#### Velocità di variazione

Quando il valore dell'Arco viene modificato tramite l'input del puntatore (dito, mouse, ecc.), la velocità di variazione è limitata in base al suo *change rate* [velocità di variazione]. La velocità di variazione è definita in gradi/secondo e può essere impostata con `lv_arc_set_change_rate(arc, rate)`

#### Offset della manopola

Modificando l'offset del "knob" [manopola] è possibile spostarne la posizione rispetto alla fine dell'Arco. L'offset della manopola può essere impostato con `lv_arc_set_knob_offset(arc, offset_angle)` e sarà visibile solo se *LV\_PART\_KNOB* è visibile.

#### Impostazione dell'indicatore tramite programmazione

È possibile impostare l'angolo dell'indicatore direttamente con una qualsiasi di queste funzioni:

- `lv_arc_set_start_angle(arc, start_angle)`
- `lv_arc_set_end_angle(arc, end_angle)`

- `lv_arc_set_angles(arc, start_angle, end_angle)`

Quando vengono utilizzati, "value" e "mode" vengono ignorati.

In altre parole, le impostazioni di angolo e valore sono indipendenti. Si consiglia di utilizzare esclusivamente uno dei due metodi. Combinarli potrebbe causare comportamenti indesiderati.

Per rendere l'arco non regolabile, rimuovere lo stile della manopola e rendere il Widget non cliccabile:

```
lv_obj_remove_style(arc, NULL, LV_PART_KNOB);
lv_obj_remove_flag(arc, LV_OBJ_FLAG_CLICKABLE);
```

### Area interattiva

Per default `LV_OBJ_FLAG_ADV_HITTEST` è disabilitato, il che significa che l'intera area dell'Arco è interattiva. Come di consueto, `lv_obj_set_ext_click_area()` può essere utilizzato per aumentare di un numero specificato di pixel l'area che risponderà all'input del puntatore (tocco, mouse, ecc.) all'esterno dell'Arco.

Se `LV_OBJ_FLAG_ADV_HITTEST` è abilitato, l'Arco sarà sensibile solo nell'intervallo tra gli angoli di sfondo iniziale e finale e sull'Arco stesso (non all'interno dell'Arco). In questo caso, `ext_click_area` fa sì che l'area sensibile si estenda sia verso l'interno che verso l'esterno. Inoltre, a ciascun angolo viene applicata una tolleranza di `lv_dpx(50)` pixel, estendendo l'intervallo di hit-test lungo la lunghezza dell'Arco.

### Posizionare un altro Widget sulla manopola

Un altro Widget può essere posizionato in base alla posizione corrente dell'Arco per seguirne il valore corrente (angolo). Per fare ciò, utilizzare `lv_arc_align_obj_to_angle(arc, widget_to_align, radius_offset)`.

Analogamente, `lv_arc_rotate_obj_to_angle(arc, widget_to_rotate, radius_offset)` può essere utilizzato per ruotare il Widget in base al valore corrente dell'Arco.

Un tipico caso d'uso è chiamare queste funzioni nell'evento `VALUE_CHANGED` dell'Arco.

#### 6.4.4 Eventi

- `LV_EVENT_VALUE_CHANGED` viene inviato quando l'Arco viene premuto/trascinato su un nuovo valore.

##### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

#### 6.4.5 Tasti

- `LV_KEY_RIGHT/UP` Aumenta il valore di uno.
- `LV_KEY_LEFT/DOWN` Diminuisce il valore di uno.

##### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

#### 6.4.6 Esempio

##### Arco semplice

```
#include "../../lv_examples.h"
#if LV_USE_ARC && LV_BUILD_EXAMPLES

static void value_changed_event_cb(lv_event_t * e);

void lv_example_arc_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());

    /*Create an Arc*/
    lv_obj_t * arc = lv_arc_create(lv_screen_active());
    lv_obj_set_size(arc, 150, 150);
    lv_arc_set_rotation(arc, 135);
    lv_arc_set_bg_angles(arc, 0, 270);
    lv_arc_set_value(arc, 10);
    lv_obj_center(arc);
    lv_obj_add_event_cb(arc, value_changed_event_cb, LV_EVENT_VALUE_CHANGED, label);

    /*Manually update the label for the first time*/
    lv_obj_send_event(arc, LV_EVENT_VALUE_CHANGED, NULL);
}

static void value_changed_event_cb(lv_event_t * e)
{
    lv_obj_t * arc = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *)lv_event_get_user_data(e);

    lv_label_set_text_fmt(label, "%" LV_PRId32 "%%", lv_arc_get_value(arc));

    /*Rotate the label to the current position of the arc*/
    lv_arc_rotate_obj_to_angle(arc, label, 25);
}

#endif
```

## Loader con Arc

```
#include "../../lv_examples.h"
#if LV_USE_ARC && LV_BUILD_EXAMPLES

static void set_angle(void * obj, int32_t v)
{
    lv_arc_set_value((lv_obj_t *)obj, v);
}

/**
 * Create an arc which acts as a loader.
 */
void lv_example_arc_2(void)
{
    /*Create an Arc*/
    lv_obj_t * arc = lv_arc_create(lv_screen_active());
    lv_arc_set_rotation(arc, 270);
    lv_arc_set_bg_angles(arc, 0, 360);
    lv_obj_remove_style(arc, NULL, LV_PART_KNOB); /*Be sure the knob is not displayed*/
    lv_obj_remove_flag(arc, LV_OBJ_FLAG_CLICKABLE); /*To not allow adjusting by click*/
    lv_obj_center(arc);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, arc);
    lv_anim_set_exec_cb(&a, set_angle);
    lv_anim_set_duration(&a, 1000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE); /*Just for the demo*/
    lv_anim_set_repeat_delay(&a, 500);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_start(&a);
}

#endif
```

## Grafico a torta con sezioni cliccabili tramite Archi

```
#include "../../lv_examples.h"
#if LV_USE_ARC && LV_BUILD_EXAMPLES

#define CHART_SIZE 150
#define SLICE_OFFSET 20

typedef struct {
    int start_angle;
    int end_angle;
    int mid_angle;
    lv_point_t home;
    bool out;
} slice_info_t;

typedef struct {
    lv_obj_t * obj;
    int start_x;
    int start_y;
    int end_x;
    int end_y;
} slice_anim_data_t;

static float angle_accum = 0.0f;
static slice_info_t * active_info = NULL;
static lv_obj_t * active_arc = NULL;

static void anim_move_cb(void * var, int32_t v)
{
```

(continues on next page)

(continua dalla pagina precedente)

```

slice_anim_data_t * d = (slice_anim_data_t *) var;
int32_t x = d->start_x + ((d->end_x - d->start_x) * v) / 100;
int32_t y = d->start_y + ((d->end_y - d->start_y) * v) / 100;
lv_obj_set_pos(d->obj, x, y);
}

static void anim_cleanup_cb(lv_anim_t * a)
{
    lv_free(a->var);
}

static void arc_click_cb(lv_event_t * e)
{
    lv_obj_t * arc = lv_event_get_target_obj(e);
    slice_info_t * info = (slice_info_t *)lv_event_get_user_data(e);

    int32_t x_off = (SLICE_OFFSET * lv_trigo_cos(info->mid_angle)) >> LV_TRIGO_SHIFT;
    int32_t y_off = (SLICE_OFFSET * lv_trigo_sin(info->mid_angle)) >> LV_TRIGO_SHIFT;

    if(active_info && active_info != info && active_info->out) {
        slice_anim_data_t * anim_back = (slice_anim_data_t *) lv_malloc(sizeof(slice_anim_data_t));
        anim_back->obj = active_arc;
        anim_back->start_x = lv_obj_get_x(active_arc) - SLICE_OFFSET;
        anim_back->start_y = lv_obj_get_y(active_arc) - SLICE_OFFSET;
        anim_back->end_x = active_info->home.x;
        anim_back->end_y = active_info->home.y;
        active_info->out = false;

        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, anim_back);
        lv_anim_set_exec_cb(&a, anim_move_cb);
        lv_anim_set_time(&a, 200);
        lv_anim_set_values(&a, 0, 100);
        lv_anim_set_deleted_cb(&a, anim_cleanup_cb);
        lv_anim_start(&a);
    }

    int target_x, target_y;
    if(info->out) {
        target_x = info->home.x;
        target_y = info->home.y;
        info->out = false;
        active_info = NULL;
        active_arc = NULL;
    }
    else {
        target_x = info->home.x + x_off;
        target_y = info->home.y + y_off;
        info->out = true;
        active_info = info;
        active_arc = arc;
    }

    slice_anim_data_t * anim_data = (slice_anim_data_t *) lv_malloc(sizeof(slice_anim_data_t));
    anim_data->obj = arc;
    anim_data->start_x = lv_obj_get_x(arc) - SLICE_OFFSET;
    anim_data->start_y = lv_obj_get_y(arc) - SLICE_OFFSET;
    anim_data->end_x = target_x;
    anim_data->end_y = target_y;

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, anim_data);
    lv_anim_set_exec_cb(&a, anim_move_cb);
    lv_anim_set_time(&a, 200);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_deleted_cb(&a, anim_cleanup_cb);
    lv_anim_start(&a);
}

static void create_slice(lv_obj_t * parent, int percentage, lv_color_t color)
{
    if(percentage <= 0) return;

    float slice_angle = (percentage * 360.0f) / 100.0f;
    int start = (int)(angle_accum + 0.5f);
    angle_accum += slice_angle;
    int end = (int)(angle_accum + 0.5f);
    if(end > 360) end = 360;

    lv_obj_t * arc = lv_arc_create(parent);
    lv_obj_set_size(arc, CHART_SIZE, CHART_SIZE);
    lv_obj_center(arc);

    lv_arc_set_mode(arc, LV_ARC_MODE_NORMAL);
    lv_arc_set_bg_start_angle(arc, start);
    lv_arc_set_bg_end_angle(arc, end);

    lv_obj_set_style_arc_width(arc, CHART_SIZE / 2, LV_PART_MAIN);
    lv_obj_set_style_arc_width(arc, 0, LV_PART_INDICATOR);

    lv_obj_set_style_arc_color(arc, color, LV_PART_MAIN);
    lv_obj_set_style_arc_rounded(arc, false, LV_PART_MAIN);
    lv_obj_remove_style(arc, NULL, LV_PART_KNOB);
    lv_obj_add_flag(arc, LV_OBJ_FLAG_ADV_HITTEST);

    lv_obj_t * label = lv_label_create(arc);
    lv_label_set_text_fmt(label, "%d%%", percentage);
    int mid_angle = start + ((end - start) / 2);
    int radius = CHART_SIZE / 4;
    int x_offset = (radius * lv_trigo_cos(mid_angle)) >> LV_TRIGO_SHIFT;
    int y_offset = (radius * lv_trigo_sin(mid_angle)) >> LV_TRIGO_SHIFT;

    lv_obj_align(label, LV_ALIGN_CENTER, x_offset, y_offset);

    slice_info_t * info = (slice_info_t *) lv_malloc(sizeof(slice_info_t));
    info->start_angle = start;
    info->end_angle = end;
    info->mid_angle = mid_angle;
}

```

(continues on next page)

(continua dalla pagina precedente)

```

info->out      = false;
info->home.x  = lv_obj_get_x(arc);
info->home.y  = lv_obj_get_y(arc);
lv_obj_add_event_cb(arc, arc_click_cb, LV_EVENT_CLICKED, info);

}

void lv_example_arc_3(void)
{
    /* Root container: flex row */
    lv_obj_t * root = lv_obj_create(lv_screen_active());
    lv_obj_set_size(root, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_center(root);
    lv_obj_set_flex_flow(root, LV_FLEX_FLOW_ROW);
    lv_obj_set_flex_align(root, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    lv_obj_set_style_pad_all(root, 0, LV_PART_MAIN);
    lv_obj_set_style_border_width(root, 0, LV_PART_MAIN);
    lv_obj_set_style_border_color(root, lv_palette_main(LV_PALETTE_RED), LV_PART_MAIN);
    lv_obj_set_style_bg_opa(root, LV_OPA_TRANSP, LV_PART_MAIN);
    lv_obj_remove_flag(root, LV_OBJ_FLAG_SCROLLABLE);

    /* Slices container */
    lv_obj_t * slices_container = lv_obj_create(root);
    lv_obj_set_size(slices_container, CHART_SIZE + 2 * SLICE_OFFSET, CHART_SIZE + 2 * SLICE_OFFSET);
    lv_obj_set_style_pad_all(slices_container, 0, LV_PART_MAIN);
    lv_obj_set_style_margin_all(slices_container, 0, LV_PART_MAIN);
    lv_obj_set_style_border_width(slices_container, 0, LV_PART_MAIN);
    lv_obj_set_style_border_color(slices_container, lv_palette_main(LV_PALETTE_BLUE), LV_PART_MAIN);
    lv_obj_set_style_bg_opa(slices_container, LV_OPA_TRANSP, LV_PART_MAIN);
    lv_obj_remove_flag(slices_container, LV_OBJ_FLAG_SCROLLABLE);

    /* Create slices */
    angle_accum = 0.0f;
    create_slice(slices_container, 12, lv_palette_main(LV_PALETTE_RED));
    create_slice(slices_container, 18, lv_palette_main(LV_PALETTE_BLUE));
    create_slice(slices_container, 26, lv_palette_main(LV_PALETTE_GREEN));
    create_slice(slices_container, 24, lv_palette_main(LV_PALETTE_ORANGE));
    create_slice(slices_container, 20, lv_palette_main(LV_PALETTE_BLUE_GREY));
}

#endif

```

## 6.4.7 API

*lv\_observer.h*

*lv\_arc.h*

*lv\_arc\_private.h*

*lv\_arclabel.h*

*lv\_arclabel\_private.h*

## 6.5 Arc Label (lv\_arclabel)

### 6.5.1 Panoramica

La Arc Label è un widget specializzato progettato per visualizzare il testo lungo un arco. Consente un posizionamento e uno stile flessibili del testo, rendendolo adatto ad applicazioni in cui il testo deve seguire un percorso curvilineo, come in indicatori, quadranti o interfacce personalizzate. Il widget supporta diverse configurazioni, tra cui l'allineamento del testo, la direzione, la regolazione del raggio e la personalizzazione del colore.

### 6.5.2 Parti e Stili

- **LV\_PART\_MAIN** Rappresenta la parte principale dell'Arc Label, incluso il percorso dell'arco e il testo visualizzato lungo di esso. L'aspetto del testo e dell'arco può essere personalizzato utilizzando le tipiche proprietà di stile del testo e dello sfondo.

### 6.5.3 Utilizzo

#### Gestione del Testo

- Il testo può essere impostato utilizzando `lv_arclabel_set_text(arclabel, "Your text")`.
- Per il testo formattato, si usa `lv_arclabel_set_text_fmt(arclabel, "Formatted %s", "text")`.
- Il testo statico può essere impostato con `lv_arclabel_set_text_static(arclabel, static_text)`, che evita l'allocazione dinamica della memoria.

## Configurazione dell'Angolo

- Impostare l'angolo iniziale dell'arco con `lv_arclabel_set_angle_start(arclabel, angle)`.
- Definire la dimensione dell'arco (ampiezza angolare) utilizzando `lv_arclabel_set_angle_size(arclabel, size)`.
- Gli angoli sono misurati in gradi, partendo dall'asse x positivo (ore 3) e aumentando in senso orario.

## Direzione

Specificare la direzione del testo lungo l'arco utilizzando `lv_arclabel_set_dir(arclabel, LV_ARCLABEL_DIR_CLOCKWISE)` o `lv_arclabel_set_dir(arclabel, LV_ARCLABEL_DIR_COUNTERCLOCKWISE)`.

- `LV_ARCLABEL_DIR_CLOCKWISE` Il testo scorre in senso orario lungo l'arco.
- `LV_ARCLABEL_DIR_COUNTERCLOCKWISE` Il testo scorre in senso antiorario lungo l'arco.

## Allineamento

Regolare l'allineamento verticale del testo con `lv_arclabel_set_text_vertical_align(arclabel, LV_ARCLABEL_TEXT_ALIGN_CENTER)`. Impostare l'allineamento orizzontale utilizzando `lv_arclabel_set_text_horizontal_align(arclabel, LV_ARCLABEL_TEXT_ALIGN_CENTER)`.

Sia l'allineamento verticale che quello orizzontale utilizzano la stessa logica.

- `LV_ARCLABEL_TEXT_ALIGN_DEFAULT` Utilizza l'allineamento di default.
- `LV_ARCLABEL_TEXT_ALIGN_LEADING` Allinea il testo al bordo anteriore dell'arco.
- `LV_ARCLABEL_TEXT_ALIGN_CENTER` Centra il testo lungo l'arco.
- `LV_ARCLABEL_TEXT_ALIGN_TRAILING` Allinea il testo al bordo posteriore dell'arco.

## Raggio e Offset del Centro

- Impostare il raggio dell'arco con `lv_arclabel_set_radius(arclabel, radius)`.
- Regolare l'offset del centro dell'arco usando `lv_arclabel_set_center_offset_x(arclabel, x)` e `lv_arclabel_set_center_offset_y(arclabel, y)`.

## Colore e Ricolorazione

Abilitare la ricolorazione del testo con `lv_arclabel_set_recolor(arclabel, true)`. Questo consente di colorare parti del testo in modo diverso utilizzando i comandi di colore incorporati nella stringa di testo.

## Comportamento Interattivo

Per default, Arc Label non è cliccabile. Per renderla interattiva, è necessario aggiungere una gestione eventi personalizzata, poiché non eredita la cliccabilità per default.

### 6.5.4 Eventi

L'Arc Label eredita principalmente gli eventi dalla classe dell'oggetto base. Non definisce eventi specifici oltre a quelli comuni a tutti i widget. È possibile associare gestori eventi personalizzati per rispondere alle interazioni, se necessario.

### 6.5.5 Tasti

L'Arc Label non definisce associazioni di tasti specifiche oltre a quelle ereditate dalla classe dell'oggetto base. La navigazione e l'interazione tramite tastiera richiederebbero un'implementazione aggiuntiva.

## 6.5.6 Esempio

### Semplice Arc Label

```
#include "../../lv_examples.h"
#if LV_USE_ARCLABEL && LV_BUILD_EXAMPLES

static const char * ARCLABEL_TEXT =
    "I'm on an #FA7C45 ARC#! Centered with #12c2E9 C##8B68E8 0##c471ed L##B654E5 0##C84AB2 R##DB417A F##f64659 U##ff8888 L# feature!\n";

void lv_example_arclabel_1(void)
{
    lv_obj_t * arclabel_inner = NULL;
    lv_obj_t * arclabel_outer = NULL;
    lv_obj_t * arclabel_slogan_1 = NULL;
    lv_obj_t * arclabel_slogan_2 = NULL;

    lv_obj_set_style_bg_color(lv_screen_active(), lv_color_black(), LV_PART_MAIN);

    arclabel_inner = lv_arclabel_create(lv_screen_active());
    lv_obj_set_size(arclabel_inner, 200, 200);
    lv_obj_set_style_text_color(arclabel_inner, lv_color_white(), LV_PART_MAIN);
    lv_arclabel_set_text_static(arclabel_inner, ARCLABEL_TEXT);
    lv_arclabel_set_angle_start(arclabel_inner, 180);
    lv_arclabel_set_radius(arclabel_inner, LV_PCT(80));
    lv_arclabel_set_recolor(arclabel_inner, true);
    lv_arclabel_set_text_vertical_align(arclabel_inner, LV_ARCLABEL_TEXT_ALIGN_TRAILING);
    lv_arclabel_set_dir(arclabel_inner, LV_ARCLABEL_DIR_COUNTER_CLOCKWISE);
    lv_arclabel_set_text_horizontal_align(arclabel_inner, LV_ARCLABEL_TEXT_ALIGN_CENTER);
    lv_obj_center(arclabel_inner);

    arclabel_outer = lv_arclabel_create(lv_screen_active());
    lv_obj_set_size(arclabel_outer, 200, 200);
    lv_obj_set_style_text_letter_space(arclabel_outer, 2, LV_PART_MAIN);
    lv_obj_set_style_text_color(arclabel_outer, lv_color_hex(0x888888), LV_PART_MAIN);
    lv_arclabel_set_angle_start(arclabel_outer, -180);
    lv_arclabel_set_text_static(arclabel_outer, ARCLABEL_TEXT);
    lv_arclabel_set_radius(arclabel_outer, LV_PCT(100));
    lv_arclabel_set_recolor(arclabel_outer, true);
    lv_arclabel_set_text_vertical_align(arclabel_outer, LV_ARCLABEL_TEXT_ALIGN_LEADING);
    lv_arclabel_set_dir(arclabel_outer, LV_ARCLABEL_DIR_CLOCKWISE);
    lv_arclabel_set_text_horizontal_align(arclabel_outer, LV_ARCLABEL_TEXT_ALIGN_CENTER);
    lv_obj_center(arclabel_outer);

    arclabel_slogan_1 = lv_arclabel_create(lv_screen_active());
    lv_obj_set_size(arclabel_slogan_1, 300, 200);
    lv_obj_set_style_text_letter_space(arclabel_slogan_1, 2, LV_PART_MAIN);
    lv_obj_set_style_text_color(arclabel_slogan_1, lv_palette_main(LV_PALETTE_AMBER), LV_PART_MAIN);
    lv_arclabel_set_text_static(arclabel_slogan_1, "STAY HUNGRY");
    lv_arclabel_set_offset(arclabel_slogan_1, 30);
    lv_arclabel_set_radius(arclabel_slogan_1, 150);
    lv_arclabel_set_recolor(arclabel_slogan_1, true);
    lv_arclabel_set_text_vertical_align(arclabel_slogan_1, LV_ARCLABEL_TEXT_ALIGN_TRAILING);
    lv_arclabel_set_text_horizontal_align(arclabel_slogan_1, LV_ARCLABEL_TEXT_ALIGN_CENTER);
    lv_arclabel_set_dir(arclabel_slogan_1, LV_ARCLABEL_DIR_COUNTER_CLOCKWISE);
    lv_obj_center(arclabel_slogan_1);

    arclabel_slogan_2 = lv_arclabel_create(lv_screen_active());
    lv_obj_set_size(arclabel_slogan_2, 300, 200);
    lv_obj_set_style_text_letter_space(arclabel_slogan_2, 2, LV_PART_MAIN);
    lv_obj_set_style_text_color(arclabel_slogan_2, lv_palette_main(LV_PALETTE_AMBER), LV_PART_MAIN);
    lv_arclabel_set_text_static(arclabel_slogan_2, "STAY FOOLISH");
    lv_arclabel_set_offset(arclabel_slogan_2, 30);
    lv_arclabel_set_radius(arclabel_slogan_2, 150);
    lv_arclabel_set_angle_start(arclabel_slogan_2, 180);
    lv_arclabel_set_recolor(arclabel_slogan_2, true);
    lv_arclabel_set_text_vertical_align(arclabel_slogan_2, LV_ARCLABEL_TEXT_ALIGN_TRAILING);
    lv_arclabel_set_text_horizontal_align(arclabel_slogan_2, LV_ARCLABEL_TEXT_ALIGN_CENTER);
    lv_arclabel_set_dir(arclabel_slogan_2, LV_ARCLABEL_DIR_COUNTER_CLOCKWISE);
    lv_obj_center(arclabel_slogan_2);

#endif
#if LV_FONT_MONTserrat_18
    lv_obj_set_style_text_font(arclabel_inner, &lv_font_montserrat_18, LV_PART_MAIN);
    lv_obj_set_style_text_font(arclabel_outer, &lv_font_montserrat_18, LV_PART_MAIN);
#endif
#if LV_FONT_MONTserrat_24
    lv_obj_set_style_text_font(arclabel_slogan_1, &lv_font_montserrat_24, LV_PART_MAIN);
    lv_obj_set_style_text_font(arclabel_slogan_2, &lv_font_montserrat_24, LV_PART_MAIN);
#endif
}

#endif
```

## 6.5.7 API

*lv\_arclabel.h*

*lv\_arclabel\_private.h*

## 6.6 Bar (lv\_bar)

### 6.6.1 Panoramica

Il Widget "Bar" [barra] ha uno sfondo e un indicatore. La lunghezza dell'indicatore rispetto allo sfondo indica il valore corrente della Bar.

È possibile impostare sia il valore iniziale che quello finale della Bar. La modifica del valore iniziale su un valore diverso dal valore minimo nel suo intervallo regola la posizione iniziale dell'indicatore.

## 6.6.2 Parti e Stili

- ***LV\_PART\_MAIN*** Lo sfondo della Bar. Utilizza le *tipiche proprietà di stile del background*. L'aggiunta di un padding rende l'indicatore più piccolo o più grande. La proprietà di stile **anim\_time** imposta il tempo di animazione se i valori sono impostati con **LV\_ANIM\_ON**.
- ***LV\_PART\_INDICATOR*** L'indicatore della Bar; utilizza anche le *tipiche proprietà di stile del background*.

## 6.6.3 Utilizzo

### Orientamento e Dimensioni

- Per orientamento, larghezza e altezza, è sufficiente impostare le proprietà di stile **width** e **height**;
- ***lv\_bar\_set\_orientation*(bar, orientation)** per sovrascrivere l'orientamento causato da **width** e **height**. I valori validi per **orientation** sono:
  - ***LV\_BAR\_ORIENTATION\_AUTO***
  - ***LV\_BAR\_ORIENTATION\_HORIZONTAL***
  - ***LV\_BAR\_ORIENTATION\_VERTICAL***

### Valore e intervallo

È possibile impostare un nuovo valore con ***lv\_bar\_set\_value*(bar, new\_value, LV\_ANIM\_ON / OFF)**. Il valore viene interpretato in un intervallo (valori minimo e massimo) che può essere modificato con ***lv\_bar\_set\_range*(bar, min, max)**. L'intervallo di default è 0..100 e la direzione di disegno di default è da sinistra a destra in modalità orizzontale e dal basso verso l'alto in modalità verticale. Se il valore minimo è maggiore del valore massimo, ad esempio 100..0, la direzione di disegno viene invertita.

Il nuovo valore in ***lv\_bar\_set\_value()*** può essere impostato con o senza animazione a seconda dell'ultimo parametro (LV\_ANIM\_ON/OFF).

### Modes

La Bar può essere in una delle seguenti modalità:

- ***LV\_BAR\_MODE\_NORMAL*** Una Bar normale come descritto sopra
- ***LV\_BAR\_MODE\_SYMMETRICAL*** Disegna l'indicatore dal valore zero al valore corrente. Richiede un valore minimo negativo e un valore massimo positivo, ad esempio [-100..100].
- ***LV\_BAR\_MODE\_RANGE*** Consente anche di impostare il valore iniziale con ***lv\_bar\_set\_start\_value*(bar, new\_value, LV\_ANIM\_ON / OFF)**. Il valore iniziale deve essere inferiore al valore finale.

## 6.6.4 Eventi

I Widget Bar non inviano eventi speciali.

### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

## 6.6.5 Tasti

I Widget Bar non elaborano *tasti*.

### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

## 6.6.6 Esempio

### Barra semplice

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

void lv_example_bar_1(void)
{
    lv_obj_t * bar1 = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar1, 200, 20);
    lv_obj_center(bar1);
    lv_bar_set_value(bar1, 70, LV_ANIM_OFF);
}

#endif
```

### Stile di una barra

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/**
 * Example of styling the bar
 */
void lv_example_bar_2(void)
{
    static lv_style_t style_bg;
    static lv_style_t style_indic;

    lv_style_init(&style_bg);
    lv_style_set_border_color(&style_bg, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style_bg, 2);
    lv_style_set_pad_all(&style_bg, 6); /*To make the indicator smaller*/
    lv_style_set_radius(&style_bg, 6);
    lv_style_set_anim_duration(&style_bg, 1000);

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_radius(&style_indic, 3);

    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_obj_remove_style_all(bar); /*To have a clean start*/
    lv_obj_add_style(bar, &style_bg, 0);
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

    lv_obj_set_size(bar, 200, 20);
    lv_obj_center(bar);
    lv_bar_set_value(bar, 100, LV_ANIM_ON);
}

#endif
```

### Indicatore di temperatura

```
#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

static void set_temp(void * bar, int32_t temp)
{
    lv_bar_set_value((lv_obj_t *)bar, temp, LV_ANIM_ON);
}

/**
 * A temperature meter example
 */
void lv_example_bar_3(void)
{
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_opa(&style_indic, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indic, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_bg_grad_color(&style_indic, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_dir(&style_indic, LV_GRAD_DIR_VER);

    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);
    lv_obj_set_size(bar, 20, 200);
    lv_obj_center(bar);
    lv_bar_set_range(bar, -20, 40);

    lv_anim_t a;
    lv_anim_init(&a);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_set_exec_cb(&a, set_temp);
lv_anim_set_duration(&a, 3000);
lv_anim_set_reverse_duration(&a, 3000);
lv_anim_set_var(&a, bar);
lv_anim_set_values(&a, -20, 40);
lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
lv_anim_start(&a);

}

#endif

```

## Motivo a strisce e valore dell'intervallo

```

#include "../../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/***
 * Bar with stripe pattern and ranged value
 */
void lv_example_bar_4(void)
{
    LV_IMAGE_DECLARE(img_skew_strip);
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_image_src(&style_indic, &img_skew_strip);
    lv_style_set_bg_image_tiled(&style_indic, true);
    lv_style_set_bg_image_opa(&style_indic, LV_OPA_30);

    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_obj_add_style(bar, &style_indic, LV_PART_INDICATOR);

    lv_obj_set_size(bar, 260, 20);
    lv_obj_center(bar);
    lv_bar_set_mode(bar, LV_BAR_MODE_RANGE);
    lv_bar_set_value(bar, 90, LV_ANIM_OFF);
    lv_bar_set_start_value(bar, 20, LV_ANIM_OFF);
}

#endif

```

## Barra con direzione di base LTR e RTL

```

#include "../../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/***
 * Bar with LTR and RTL base direction
 */
void lv_example_bar_5(void)
{
    lv_obj_t * label;

    lv_obj_t * bar_ltr = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar_ltr, 200, 20);
    lv_bar_set_value(bar_ltr, 70, LV_ANIM_OFF);
    lv_obj_align(bar_ltr, LV_ALIGN_CENTER, 0, -30);

    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Left to Right base direction");
    lv_obj_align_to(label, bar_ltr, LV_ALIGN_OUT_TOP_MID, 0, -5);

    lv_obj_t * bar_rtl = lv_bar_create(lv_screen_active());
    lv_obj_set_base_dir(bar_rtl, LV_BASE_DIR_RTL, 0);
    lv_obj_set_size(bar_rtl, 200, 20);
    lv_bar_set_value(bar_rtl, 70, LV_ANIM_OFF);
    lv_obj_align(bar_rtl, LV_ALIGN_CENTER, 0, 30);

    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Right to Left base direction");
    lv_obj_align_to(label, bar_rtl, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif

```

## Cassetto personalizzato per visualizzare il valore corrente

```

#include "../../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

#define MAX_VALUE 100
#define MIN_VALUE 0

static void set_value(void * bar, int32_t v)
{
    lv_bar_set_value((lv_obj_t *)bar, v, LV_ANIM_OFF);
}

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.font = LV_FONT_DEFAULT;

    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d", (int)lv_bar_get_value(obj));

    lv_text_attributes_t attributes = {0};

```

(continues on next page)

(continua dalla pagina precedente)

```

attributes.letter_space = label_dsc.letter_space;
attributes.line_space = label_dsc.line_space;
attributes.max_width = LV_COORD_MAX;
attributes.text_flags = label_dsc.flag;

lv_point_t txt_size;
lv_text_get_size(&txt_size, buf, label_dsc.font, &attributes);

lv_area_t txt_area;
txt_area.x1 = 0;
txt_area.x2 = txt_size.x - 1;
txt_area.y1 = 0;
txt_area.y2 = txt_size.y - 1;

lv_area_t indic_area;
lv_obj_get_coords(obj, &indic_area);
lv_area_set_width(&indic_area, lv_area_get_width(&indic_area) * lv_bar_get_value(obj) / MAX_VALUE);

/*If the indicator is long enough put the text inside on the right*/
if(lv_area_get_width(&indic_area) > txt_size.x + 20) {
    lv_area_align(&indic_area, &txt_area, LV_ALIGN_RIGHT_MID, -10, 0);
    label_dsc.color = lv_color_white();
}
/*If the indicator is still short put the text out of it on the right*/
else {
    lv_area_align(&indic_area, &txt_area, LV_ALIGN_OUT_RIGHT_MID, 10, 0);
    label_dsc.color = lv_color_black();
}
label_dsc.text = buf;
label_dsc.text_local = true;
lv_layer_t * layer = lv_event_get_layer(e);
lv_draw_label(layer, &label_dsc, &txt_area);
}

/***
 * Custom drawer on the bar to display the current value
 */
void lv_example_bar_6(void)
{
    lv_obj_t * bar = lv_bar_create(lv_screen_active());
    lv_bar_set_range(bar, MIN_VALUE, MAX_VALUE);
    lv_obj_set_size(bar, 200, 20);
    lv_obj_center(bar);
    lv_obj_add_event_cb(bar, event_cb, LV_EVENT_DRAW_MAIN_END, NULL);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, bar);
    lv_anim_set_values(&a, 0, 100);
    lv_anim_set_exec_cb(&a, set_value);
    lv_anim_set_duration(&a, 4000);
    lv_anim_set_reverse_duration(&a, 4000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}
#endif

```

## Barra con direzione opposta

```

#include "../../lv_examples.h"
#if LV_USE_BAR && LV_BUILD_EXAMPLES

/***
 * Bar with opposite direction
 */
void lv_example_bar_7(void)
{
    lv_obj_t * bar_tob = lv_bar_create(lv_screen_active());
    lv_obj_set_size(bar_tob, 20, 180);
    lv_bar_set_range(bar_tob, 100, 0);
    lv_bar_set_value(bar_tob, 70, LV_ANIM_OFF);
    lv_obj_align(bar_tob, LV_ALIGN_CENTER, 0, 10);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "From top to bottom");
    lv_obj_align_to(label, bar_tob, LV_ALIGN_OUT_TOP_MID, 0, -5);
}
#endif

```

## 6.6.7 API

*lv\_barcode.h*

*lv\_barcode\_private.h*

*lv\_bar.h*

*lv\_bar\_private.h*

## 6.7 Button (lv\_button)

### 6.7.1 Panoramica

I "Button" [pulsanti] non presentano nuove funzionalità rispetto al [Widget Base \(lv\\_obj\)](#). Sono utili per scopi semantici e hanno impostazioni di default leggermente diverse.

I pulsanti, per default, differiscono dal Widget Base nei seguenti modi:

- Non "scrollable"
- Aggiunti al gruppo di default
- Altezza e larghezza di default impostate su `LV_SIZE_CONTENT`

### 6.7.2 Parti e Stili

- `LV_PART_MAIN` Lo sfondo del pulsante; utilizza le *tipiche proprietà di stile del background*.

### 6.7.3 Utilizzo

Non ci sono nuove funzionalità rispetto al [Base Widget](#).

### 6.7.4 Eventi

- `LV_EVENT_VALUE_CHANGED` quando il flag `LV_OBJ_FLAG_CHECKABLE` è abilitato e si clicca sul Widget. L'evento si verifica durante la transizione da/verso lo stato selezionato.

#### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

### 6.7.5 Tasti

Si noti che lo stato di `LV_KEY_ENTER` viene tradotto in `LV_EVENT_PRESSED`, `LV_EVENT_PRESSING` e `LV_EVENT_RELEASED` ecc.

#### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

### 6.7.6 Esempio

#### Pulsanti semplici

```
#include "../../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);

    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked");
    } else if(code == LV_EVENT_VALUE_CHANGED) {
        LV_LOG_USER("Toggled");
    }
}

void lv_example_button_1(void)
{
    lv_obj_t * label;

    lv_obj_t * btn1 = lv_button_create(lv_screen_active());
    lv_obj_add_event_cb(btn1, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -40);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_remove_flag(btn1, LV_OBJ_FLAG_PRESS_LOCK);

label = lv_label_create(btn1);
lv_label_set_text(label, "Button");
lv_obj_center(label);

lv_obj_t * btn2 = lv_button_create(lv_screen_active());
lv_obj_add_event_cb(btn2, event_handler, LV_EVENT_ALL, NULL);
lv_obj_align(btn2, LV_ALIGN_CENTER, 0, 40);
lv_obj_add_flag(btn2, LV_OBJ_FLAG_CHECKABLE);
lv_obj_set_height(btn2, LV_SIZE_CONTENT);

label = lv_label_create(btn2);
lv_label_set_text(label, "Toggle");
lv_obj_center(label);

}

#endif

```

## Pulsanti con stili

```

#include "../../../lv_examples.h"
#if LV_USE_BUTTON && LV_BUILD_EXAMPLES

/***
 * Style a button from scratch
 */
void lv_example_button_2(void)
{
    /*Init the style for the default state*/
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 3);

    lv_style_set_bg_opa(&style, LV_OPA_100);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_bg_grad_color(&style, lv_palette_darken(LV_PALETTE_BLUE, 2));
    lv_style_set_bg_grad_dir(&style, LV_GRAD_DIR_VER);

    lv_style_set_border_opa(&style, LV_OPA_40);
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_GREY));

    lv_style_set_shadow_width(&style, 8);
    lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_shadow_offset_y(&style, 8);

    lv_style_set_outline_opa(&style, LV_OPA_COVER);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    lv_style_set_text_color(&style, lv_color_white());
    lv_style_set_pad_all(&style, 10);

    /*Init the pressed style*/
    static lv_style_t style_pr;
    lv_style_init(&style_pr);

    /*Add a large outline when pressed*/
    lv_style_set_outline_width(&style_pr, 30);
    lv_style_set_outline_opa(&style_pr, LV_OPA_TRANSP);

    lv_style_set_translate_y(&style_pr, 5);
    lv_style_set_shadow_offset_y(&style_pr, 3);
    lv_style_set_bg_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 2));
    lv_style_set_bg_grad_color(&style_pr, lv_palette_darken(LV_PALETTE_BLUE, 4));

    /*Add a transition to the outline*/
    static lv_style_transition_dsc_t trans;
    static lv_style_prop_t props[] = {LV_STYLE_OUTLINE_WIDTH, LV_STYLE_OUTLINE_OPA, 0};
    lv_style_transition_dsc_init(&trans, props, lv_anim_path_linear, 300, 0, NULL);

    lv_style_set_transition(&style_pr, &trans);

    lv_obj_t * btn1 = lv_button_create(lv_screen_active());
    lv_obj_remove_style_all(btn1); /*Remove the style coming from the theme*/
    lv_obj_add_style(btn1, &style_pr, LV_STATE_PRESSED);
    lv_obj_set_size(btn1, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_obj_center(btn1);

    lv_obj_t * label = lv_label_create(btn1);
    lv_label_set_text(label, "Button");
    lv_obj_center(label);
}
#endif

```

## Pulsante gommoso

```

#include "../../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_BUTTON

/***
 * Create a style transition on a button to act like a gum when clicked
 */
void lv_example_button_3(void)
{
    /*Properties to transition*/
    static lv_style_prop_t props[] = {
        LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_TRANSFORM_HEIGHT, LV_STYLE_TEXT_LETTER_SPACE, 0
    };

    /*Transition descriptor when going back to the default state.
     *Add some delay to be sure the press transition is visible even if the press was very short*/

```

(continues on next page)

(continua dalla pagina precedente)

```

static lv_style_transition_dsc_t transition_dsc_def;
lv_style_transition_dsc_init(&transition_dsc_def, props, lv_anim_path_overshoot, 250, 100, NULL);

/*Transition descriptor when going to pressed state.
 *No delay, go to presses state immediately*/
static lv_style_transition_dsc_t transition_dsc_pr;
lv_style_transition_dsc_init(&transition_dsc_pr, props, lv_anim_path_ease_in_out, 250, 0, NULL);

/*Add only the new transition to he default state*/
static lv_style_t style_def;
lv_style_init(&style_def);
lv_style_set_transition(&style_def, &transition_dsc_def);

/*Add the transition and some transformation to the presses state.*/
static lv_style_t style_pr;
lv_style_init(&style_pr);
lv_style_set_transform_width(&style_pr, 10);
lv_style_set_transform_height(&style_pr, -10);
lv_style_set_text_letter_space(&style_pr, 10);
lv_style_set_transition(&style_pr, &transition_dsc_pr);

lv_obj_t * btn1 = lv_button_create(lv_screen_active());
lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -80);
lv_obj_add_style(btn1, &style_pr, LV_STATE_PRESSED);
lv_obj_add_style(btn1, &style_def, 0);

lv_obj_t * label = lv_label_create(btn1);
lv_label_set_text(label, "Gum");
}
#endif

```

## 6.7.7 API

*lv\_api\_map\_v9\_0.h*

*lv\_button.h*

*lv\_button\_private.h*

*lv\_buttonmatrix.h*

*lv\_buttonmatrix\_private.h*

## 6.8 Button Matrix (*lv\_buttonmatrix*)

### 6.8.1 Panoramica

Il Widget "Button Matrix" [Matrice Pulsanti] è un modo semplice per visualizzare più pulsanti in righe e colonne, semplice perché i pulsanti non vengono effettivamente creati, ma solo disegnati virtualmente al volo. Con la Button Matrix, ogni pulsante utilizza solo otto byte di memoria aggiuntivi invece dei circa 100-150 byte di un normale Widget *Button* più i circa 100 byte per il Widget *Label*.

I nuovi Widget Button Matrix vengono aggiunti al gruppo di default (se impostato). Inoltre, Button Matrix un Widget modificabile: consente di selezionare e cliccare sui pulsanti anche con encoder e navigazione da tastiera.

### 6.8.2 Parti e Stili

- *LV\_PART\_MAIN* Il background della Button Matrix, utilizza le tipiche proprietà di stile del background. *pad\_row* e *pad\_column* impostano lo spazio tra i pulsanti.
- *LV\_PART\_ITEMS* Tutti i pulsanti utilizzano le tipiche proprietà di stile del testo e dello sfondo, ad eccezione di traslazioni e trasformazioni.

### 6.8.3 Utilizzo

#### Mappa dei pulsanti

Il numero di pulsanti, le loro posizioni e il testo sono controllati da un array di stringhe descrittive, chiamato *map*, passato a *lv\_buttonmatrix\_set\_map*(*btn\_matrix*, *my\_map*). La dichiarazione di una map dovrebbe essere simile a `const char * map[] = {"button1", "button2", "button3", NULL};`. Si noti che l'ultimo elemento deve essere `NULL` o una stringa vuota ("")!

Utilizzare "\n" nella map per inserire un **line break** [andata-a-capo]. Ad esempio `` {"button1", "button2", "\n", "button3", ""}``. La larghezza dei pulsanti di ogni riga viene calcolata automaticamente.

mente. Quindi, nell'esempio, la prima riga avrà 2 pulsanti, ciascuno con larghezza pari al 50%, e una seconda riga con 1 pulsante con larghezza pari al 100%.

### Nota

Il numero di pulsanti non include né gli elementi di "newline" né l'elemento di chiusura dell'array.

## Larghezza dei pulsanti

La larghezza dei pulsanti può essere impostata in proporzione alla larghezza degli altri pulsanti nella stessa riga con `lv_buttonmatrix_set_button_width(btn_matrix, button_id, width)`. Ad esempio, in una riga con due pulsanti: `buttonA, width = 1` e `buttonB, width = 2`, `buttonA` avrà una larghezza pari al 33% e `buttonB` una larghezza pari al 66%. Questo è simile al funzionamento della proprietà "flex-grow" in CSS. La larghezza deve essere compresa tra [1 e 15], con il valore di default pari a 1.

## Comportamento dei pulsanti

Il comportamento di ogni pulsante può essere personalizzato con i seguenti flag di controllo:

- `LV_BUTTONMATRIX_CTRL_HIDDEN`: Nasconde il pulsante; continua a occupare il suo spazio nel layout.
- `LV_BUTTONMATRIX_CTRL_NO_REPEAT`: Non emette eventi `LV_EVENT_LONG_PRESSED_REPEAT` mentre il pulsante è premuto a lungo.
- `LV_BUTTONMATRIX_CTRL_DISABLED`: Disabilita il pulsante come `LV_STATE_DISABLED` nei Widget normali.
- `LV_BUTTONMATRIX_CTRL_CHECKABLE`: Abilita l'attivazione/disattivazione di `LV_STATE_CHECKED` quando cliccato.
- `LV_BUTTONMATRIX_CTRL_CHECKED`: Imposta il pulsante come selezionato. Utilizzerà gli stili `LV_STATE_CHECKED`.
- `LV_BUTTONMATRIX_CTRL_CLICK_TRIG`: 1: Abilita l'invio di `LV_EVENT_VALUE_CHANGE` al CLICK, 0: invia `LV_EVENT_VALUE_CHANGE` al PRESS.
- `LV_BUTTONMATRIX_CTRL_POPOVER`: Mostra il testo del pulsante in un pop-over quando viene premuto.
- `LV_BUTTONMATRIX_CTRL_RECOLOR`: Abilita la ricolorazione del testo con `#color`
- `LV_BUTTONMATRIX_CTRL_CUSTOM_1`: Flag personalizzato di libero utilizzo
- `LV_BUTTONMATRIX_CTRL_CUSTOM_2`: Flag personalizzato di libero utilizzo

Per default, questi flag sono disabilitati.

Per impostare e cancellare i flag di controllo di un pulsante, utilizzare

- `lv_buttonmatrix_set_button_ctrl(btn_matrix, button_id, LV_BUTTONMATRIX_CTRL_...)` e
- `lv_buttonmatrix_clear_button_ctrl(btn_matrix, button_id, LV_BUTTONMATRIX_CTRL_...)`

rispettivamente. `button_id` è un indice del pulsante a base zero (0 = primo pulsante). I valori di `LV_BUTTONMATRIX_CTRL_...` possono essere combinati tramite OR bit a bit quando vengono passati a queste funzioni.

Per impostare e cancellare lo stesso attributo di controllo per tutti i pulsanti in una Button Matrix, usare

- `lv_buttonmatrix_set_button_ctrl_all(btn_matrix, LV_BUTTONMATRIX_CTRL_...)` e
- `lv_buttonmatrix_clear_button_ctrl_all(btn_matrix, LV_BUTTONMATRIX_CTRL_...)`

rispettivamente.

Per impostare una mappa di controllo per una Button Matrix (simile a [Button map](#)), si usa

- `lv_buttonmatrix_set_ctrl_map(btn_matrix, ctrl_map)`.

Un elemento di `ctrl_map` dovrebbe apparire come `ctrl_map[0] = width | LV_BUTTONMATRIX_CTRL_NO_REPEAT | LV_BUTTONMATRIX_CTRL_CHECKABLE`. Il numero di elementi deve essere uguale al numero di pulsanti.

#### Uno selezionato

La funzione "One-checked" [Uno selezionato] può essere abilitata con `lv_buttonmatrix_set_one_checked(btn_matrix, true)` per consentire la selezione di un solo pulsante alla volta.

#### 6.8.4 Eventi

- `LV_EVENT_VALUE_CHANGED`: Inviato quando un pulsante viene premuto/rilasciato o ripetuto dopo una pressione prolungata. Il parametro event è impostato sull'ID del pulsante premuto/rilasciato.

`lv_buttonmatrix_get_selected_button(btn_matrix)` restituisce l'indice del pulsante rilasciato più di recente (il pulsante con il focus) oppure `LV_BUTTONMATRIX_BUTTON_NONE` se non è stato trovato alcun pulsante di questo tipo.

`lv_buttonmatrix_get_button_text(btn_matrix, button_id)` restituisce un puntatore al testo del pulsante specificato dall'indice a base zero `button_id`.

##### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

#### 6.8.5 Tasti

- `LV_KEY_RIGHT/UP/LEFT/RIGHT` Per navigare tra i pulsanti e selezionarne uno
- `LV_KEY_ENTER` Per premere/rilasciare il pulsante selezionato

Si noti che una pressione prolungata sulla Button Matrix con un encoder può significare entrare/uscire dalla modalità di modifica e che una semplice pressione prolungata di un pulsante può anche significare ripeterlo. Per evitare questa contraddizione, aggiungere `lv_buttonmatrix_set_button_ctrl_all(btn_matrix, LV_BUTTONMATRIX_CTRL_CLICK_TRIG | LV_BUTTONMATRIX_CTRL_NO_REPEAT)` alla Button Matrix se utilizzata con un encoder. In questo modo la funzione di ripetizione viene disattivata, quindi il pulsante non verrà attivato quando si esce dalla modalità di modifica.

##### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

#### 6.8.6 Esempio

##### Matrice di pulsanti semplice

```
#include "../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        uint32_t id = lv_buttonmatrix_get_selected_button(obj);
        const char * txt = lv_buttonmatrix_get_button_text(obj, id);
        LV_UNUSED(txt);
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

    LV_LOG_USER("%s was pressed\n", txt);
}

static const char * bnm_map[] = {"1", "2", "3", "4", "5", "\n",
                                "6", "7", "8", "9", "0", "\n",
                                "Action1", "Action2", ""};

void lv_example_buttonmatrix_1(void)
{
    lv_obj_t * bnm1 = lv_buttonmatrix_create(lv_screen_active());
    lv_buttonmatrix_set_map(bnm1, bnm_map);
    lv_buttonmatrix_set_button_width(bnm1, 10, 2); /*Make "Action1" twice as wide as "Action2"*/
    lv_buttonmatrix_set_button_ctrl(bnm1, 10, LV_BUTTONMATRIX_CTRL_CHECKABLE);
    lv_buttonmatrix_set_button_ctrl(bnm1, 11, LV_BUTTONMATRIX_CTRL_CHECKED);
    lv_obj_align(bnm1, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(bnm1, event_handler, LV_EVENT_ALL, NULL);
}

#endif

```

## Pulsanti personalizzati

```

#include "../../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    /*When the button matrix draws the buttons...*/
    if(base_dsc->part == LV_PART_ITEMS) {
        bool pressed = false;
        if(lv_buttonmatrix_get_selected_button(obj) == base_dsc->idl && lv_obj_has_state(obj, LV_STATE_PRESSED)) {
            pressed = true;
        }

        /*Change the draw descriptor of the 2nd button*/
        if(base_dsc->idl == 1) {
            lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
            if(fill_draw_dsc) {
                fill_draw_dsc->radius = 0;
                if(pressed) fill_draw_dsc->color = lv_palette_darken(LV_PALETTE_BLUE, 3);
                else fill_draw_dsc->color = lv_palette_main(LV_PALETTE_BLUE);
            }
            lv_draw_box_shadow_dsc_t * box_shadow_draw_dsc = lv_draw_task_get_box_shadow_dsc(draw_task);
            if(box_shadow_draw_dsc) {
                box_shadow_draw_dsc->width = 6;
                box_shadow_draw_dsc->ofs_x = 3;
                box_shadow_draw_dsc->ofs_y = 3;
            }
            lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
            if(label_draw_dsc) {
                label_draw_dsc->color = lv_color_white();
            }
        }

        /*Change the draw descriptor of the 3rd button*/
        else if(base_dsc->idl == 2) {
            lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
            if(fill_draw_dsc) {
                fill_draw_dsc->radius = LV_RADIUS_CIRCLE;
                if(pressed) fill_draw_dsc->color = lv_palette_darken(LV_PALETTE_RED, 3);
                else fill_draw_dsc->color = lv_palette_main(LV_PALETTE_RED);
            }

            lv_draw_box_shadow_dsc_t * box_shadow_draw_dsc = lv_draw_task_get_box_shadow_dsc(draw_task);
            if(box_shadow_draw_dsc) {
                box_shadow_draw_dsc->radius = LV_RADIUS_CIRCLE;
            }
        }
        else if(base_dsc->idl == 3) {
            lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
            if(label_draw_dsc) {
                label_draw_dsc->opa = 0;
            }
            if(lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_FILL) {
                LV_IMAGE_DECLARE(img_star);
                lv_image_header_t header;
                lv_result_t res = lv_image_decoder_get_info(&img_star, &header);
                if(res != LV_RESULT_OK) return;

                lv_area_t a;
                a.x1 = 0;
                a.x2 = header.w - 1;
                a.y1 = 0;
                a.y2 = header.h - 1;
                lv_area_t draw_task_area;
                lv_draw_task_get_area(draw_task, &draw_task_area);
                lv_area_align(&draw_task_area, &a, LV_ALIGN_CENTER, 0, 0);

                lv_draw_image_dsc_t img_draw_dsc;
                lv_draw_image_dsc_init(&img_draw_dsc);
                img_draw_dsc.src = &img_star;
                img_draw_dsc.recolor = lv_color_black();
                if(pressed) img_draw_dsc.recolor_opa = LV_OPA_30;

                lv_draw_image(base_dsc->layer, &img_draw_dsc, &a);
            }
        }
    }
}

/**
 * Add custom drawer to the button matrix to customize buttons one by one

```

(continues on next page)

(continua dalla pagina precedente)

```
/*
void lv_example_buttonmatrix_2(void)
{
    lv_obj_t * btm = lv_buttonmatrix_create(lv_screen_active());
    lv_obj_add_event_cb(btm, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(btm, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_obj_center(btm);
}

#endif
```

## Paginazione

```
#include "../../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    uint32_t id = lv_buttonmatrix_get_selected_button(obj);
    bool prev = id == 0;
    bool next = id == 6;
    if(prev || next) {
        /*Find the checked button*/
        uint32_t i;
        for(i = 1; i < 7; i++) {
            if(lv_buttonmatrix_has_button_ctrl(obj, i, LV_BUTTONMATRIX_CTRL_CHECKED)) break;
        }

        if(prev && i > 1) i--;
        else if(next && i < 5) i++;

        lv_buttonmatrix_set_button_ctrl(obj, i, LV_BUTTONMATRIX_CTRL_CHECKED);
    }
}

/**
 * Make a button group (pagination)
 */
void lv_example_buttonmatrix_3(void)
{
    static lv_style_t style_bg;
    lv_style_init(&style_bg);
    lv_style_set_pad_all(&style_bg, 0);
    lv_style_set_pad_gap(&style_bg, 0);
    lv_style_set_clip_corner(&style_bg, true);
    lv_style_set_radius(&style_bg, LV_RADIUS_CIRCLE);
    lv_style_set_border_width(&style_bg, 0);

    static lv_style_t style_btn;
    lv_style_init(&style_btn);
    lv_style_set_radius(&style_btn, 0);
    lv_style_set_border_width(&style_btn, 1);
    lv_style_set_border_opa(&style_btn, LV_OPA_50);
    lv_style_set_border_color(&style_btn, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_border_side(&style_btn, LV_BORDER_SIDE_INTERNAL);
    lv_style_set_radius(&style_btn, 0);

    static const char * map[] = {LV_SYMBOL_LEFT, "1", "2", "3", "4", "5", LV_SYMBOL_RIGHT, ""};

    lv_obj_t * btm = lv_buttonmatrix_create(lv_screen_active());
    lv_buttonmatrix_set_map(btm, map);
    lv_obj_add_style(btm, &style_bg, 0);
    lv_obj_add_style(btm, &style_btn, LV_PART_ITEMS);
    lv_obj_add_event_cb(btm, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_size(btm, 225, 35);

    /*Allow selecting on one number at time*/
    lv_buttonmatrix_set_button_ctrl_all(btm, LV_BUTTONMATRIX_CTRL_CHECKABLE);
    lv_buttonmatrix_clear_button_ctrl(btm, 0, LV_BUTTONMATRIX_CTRL_CHECKABLE);
    lv_buttonmatrix_clear_button_ctrl(btm, 6, LV_BUTTONMATRIX_CTRL_CHECKABLE);

    lv_buttonmatrix_set_one_checked(btm, true);
    lv_buttonmatrix_set_button_ctrl(btm, 1, LV_BUTTONMATRIX_CTRL_CHECKED);

    lv_obj_center(btm);
}
```

## 6.8.7 API

*lv\_buttonmatrix.h*

*lv\_buttonmatrix\_private.h*

## 6.9 Calendar (lv\_calendar)

### 6.9.1 Panoramica

Il Widget "Calendar" è un calendario classico che può:

- mostrare i giorni di qualsiasi mese in una matrice 7x7;

- mostrare il nome dei giorni;
- evidenziare il giorno corrente (oggi);
- evidenziare eventuali date definite dall'utente.

Il Calendar viene aggiunto al gruppo di default (se impostato). Il Calendar è un Widget modificabile che consente di selezionare e cliccare sulle date con la navigazione tramite encoder o tastiera, nonché con dispositivi di input di tipo puntatore.

Per rendere il Calendar flessibile, per default non mostra l'anno o il mese corrente. Al suo posto, sono disponibili delle "intestazioni" opzionali che possono essere allegate al calendario.

## 6.9.2 Parti e Stili

Il Widget Calendar utilizza il Widget *Button Matrix* per organizzare i giorni in una matrice.

- *LV\_PART\_MAIN* Sfondo del Calendar. Utilizza le *tipiche proprietà di stile del background*.
- *LV\_PART\_ITEMS* Si riferisce a date e nomi dei giorni. I flag di controllo del "Button matrix" sono impostati per differenziare i pulsanti e viene aggiunto un evento personalizzato per modificare le proprietà dei pulsanti come segue:
  - I nomi dei giorni non hanno bordo, non hanno sfondo e sono visualizzati in grigio.
  - I giorni del mese precedente e successivo hanno il flag *LV\_BUTTONMATRIX\_CTRL\_DISABLED*
  - "today" [oggi] ha un bordo più spesso con il colore primario del tema; i giorni evidenziati hanno una certa opacità con il colore primario del tema.

## 6.9.3 Utilizzo

Alcune funzioni utilizzano il tipo *lv\_calendar\_date\_t* una struttura con i campi `year`, `month` e `day`.

### Data corrente

Per impostare la data corrente (oggi), utilizzare la funzione *lv\_calendar\_set\_today\_date(calendar, year, month, day)*. `month` deve essere compreso tra 1 e 12 e `day` tra 1 e 31.

### Mese visualizzato

Per impostare la data visualizzata, si usa *lv\_calendar\_set\_month\_shown(calendar, year, month)*

### Giorni evidenziati

L'elenco delle date evidenziate deve essere memorizzato in un array *lv\_calendar\_date\_t* e applicato al Calendario chiamando *lv\_calendar\_set\_highlighted\_dates(calendar, highlighted\_dates, date\_num)*. Verrà salvato solo il puntatore dell'array, quindi l'array deve avere un ambito statico o globale.

### Nomi dei giorni

I nomi dei giorni possono essere modificati con *lv\_calendar\_set\_day\_names(calendar, day\_names)* dove `day_names` è simile a `const char * day_names[7] = {"Su", "Mo", ...};`; Viene salvato solo il puntatore dei nomi dei giorni, quindi l'array dovrebbe avere un ambito statico o globale.

### Elenco anni personalizzato

Impostare un elenco anni personalizzato con *lv\_calendar\_header\_dropdown\_set\_year\_list(calendar, years\_list)* dove `years_list` è un puntatore all'elenco anni personalizzato. Può essere una stringa costante come `static const char * years = "2023\n2022\n2021\n2020\n2019";`, oppure può essere generato dinamicamente in un buffer. Il calendario memorizza questi dati in un Widget "Drop-Down List" tramite *lv\_dropdown\_set\_options()*, in modo che il puntatore stringa passato possa essere fornito da una variabile locale o da un buffer e non debba persistere oltre la chiamata.

## Calendario cinese

Il calendario cinese è uno strumento culturale tradizionale che integra elementi come il calendario lunare, i termini solari e le festività tradizionali. È ampiamente utilizzato nella vita sociale cinese, aiutando le persone a comprendere le date del calendario lunare, a organizzare le attività festive e a ereditare l'eccellente cultura tradizionale della nazione cinese. Che si tratti di famiglia, lavoro o istruzione, il calendario cinese svolge un ruolo insostituibile, consentendo alle persone di comprendere e apprezzare meglio il fascino della cultura tradizionale cinese.

Per utilizzare il calendario cinese, si usa `lv_calendar_set_chinese_mode(calendar, true)` per abilitarlo.

### 6.9.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato se si clicca su una data. `lv_calendar_get_pressed_date(calendar, &date)` per impostare `date` sulla data attualmente premuta. Restituisce `LV_RESULT_OK` se è presente una data premuta valida; in caso contrario restituisce `LV_RESULT_INVALID`.

#### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

### 6.9.5 Tasti

- `LV_KEY_DOWN/UP/LEFT/RIGHT` Per navigare tra i pulsanti fino alle date
- `LV_KEY_ENTER` Per premere/rilasciare la data selezionata

#### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

### 6.9.6 Intestazioni

Da LVGL v8.1 in poi, l'intestazione viene aggiunta direttamente al widget Calendar e l'API delle intestazioni è stata modificata.

#### Pulsanti freccia

`lv_calendar_add_header_arrow(calendar)` crea un'intestazione che contiene una freccia sinistra e una freccia destra ai lati e un testo tra le frecce che mostra l'anno e il mese correnti.

#### Drop-down

`lv_calendar_add_header_dropdown(calendar)` crea un'intestazione che contiene 2 Widget "Drop-Drown List" [elenco a discesa] per l'anno e il mese.

### 6.9.7 Esempio

#### Calendario con intestazione

```
#include "../../lv_examples.h"
#if LV_USE_CALENDAR && LV_BUILD_EXAMPLES
static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = (lv_obj_t *)lv_event_get_current_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_calendar_date_t date;
        if(lv_calendar_get_pressed_date(obj, &date)) {
            LV_LOG_USER("Clicked date: %02d.%02d.%d", date.day, date.month, date.year);
        }
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

    }
}

void lv_example_calendar_1(void)
{
    lv_obj_t * calendar = lv_calendar_create(lv_screen_active());
    lv_obj_set_size(calendar, 185, 230);
    lv_obj_align(calendar, LV_ALIGN_CENTER, 0, 27);
    lv_obj_add_event_cb(calendar, event_handler, LV_EVENT_ALL, NULL);

    lv_calendar_set_today_date(calendar, 2021, 02, 23);
    lv_calendar_set_monthShown(calendar, 2021, 02);

    /*Highlight a few days*/
    static lv_calendar_date_t highlighted_days[3]; /*Only its pointer will be saved so should be static*/
    highlighted_days[0].year = 2021;
    highlighted_days[0].month = 02;
    highlighted_days[0].day = 6;

    highlighted_days[1].year = 2021;
    highlighted_days[1].month = 02;
    highlighted_days[1].day = 11;

    highlighted_days[2].year = 2022;
    highlighted_days[2].month = 02;
    highlighted_days[2].day = 22;

    lv_calendar_set_highlighted_dates(calendar, highlighted_days, 3);

#if LV_USE_CALENDAR_HEADER_DROPDOWN
    lv_calendar_add_header_dropdown(calendar);
#elif LV_USE_CALENDAR_HEADER_ARROW
    lv_calendar_add_header_arrow(calendar);
#endif
}
#endif

```

## Calendario cinese

```

#include "../../lv_examples.h"
#if LV_USE_CALENDAR && LV_USE_CALENDAR_CHINESE && LV_BUILD_EXAMPLES

void lv_example_calendar_2(void)
{
    lv_obj_t * calendar = lv_calendar_create(lv_screen_active());
    lv_obj_set_size(calendar, 300, 300);
    lv_obj_align(calendar, LV_ALIGN_TOP_MID, 0, 0);

    lv_calendar_set_today_date(calendar, 2024, 03, 22);
    lv_calendar_set_monthShown(calendar, 2024, 03);

#if LV_USE_CALENDAR_HEADER_DROPDOWN
    lv_calendar_add_header_dropdown(calendar);
#elif LV_USE_CALENDAR_HEADER_ARROW
    lv_calendar_add_header_arrow(calendar);
#endif

    lv_calendar_set_chinese_mode(calendar, true);
    lv_obj_set_style_text_font(calendar, &lv_font_source_han_sans_sc_14_cjk, LV_PART_MAIN);
}

#else

void lv_example_calendar_2(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "chinese calendar is not enabled");
    lv_obj_center(label);
}

#endif

```

## 6.9.8 API

*lv\_api\_map\_v9\_1.h*  
*lv\_calendar.h*  
*lv\_calendar\_chinese.h*  
*lv\_calendar\_header\_arrow.h*  
*lv\_calendar\_header\_dropdown.h*  
*lv\_calendar\_private.h*

## 6.10 Canvas (lv\_canvas)

### 6.10.1 Panoramica

Un Canvas eredita da [Image](#) e lo estende, consentendo all'utente di disegnare qualsiasi cosa. Rettangoli, testo, immagini, linee, archi, ecc. possono essere disegnati qui utilizzando l'ampio motore di disegno di LVGL.

### 6.10.2 Parti e Stili

- [LV\\_PART\\_MAIN](#) Utilizza le proprietà *tipiche del background* e lo stile dell'immagine.

### 6.10.3 Utilizzo

#### Buffer

Il Canvas necessita di un buffer in cui memorizzare l'immagine disegnata. Per assegnare un buffer a un Canvas, si usa [lv\\_canvas\\_set\\_buffer\(canvas, buffer, width, height, LV\\_COLOR\\_FORMAT\\_...\)](#). Dove **buffer** è un buffer statico (non una semplice variabile locale) per contenere l'immagine del Canvas. Ad esempio, per un buffer ARGB8888 100x50: `static uint8_t buffer[100 * 50 * 4]`.

Oppure si può usare `static uint8_t buffer[LV_CANVAS_BUF_SIZE(width, height, bits_per_pixel, stride_in_bytes)]`.

Il Canvas supporta tutti i formati colore come [LV\\_COLOR\\_FORMAT\\_ARGB8888](#) o [LV\\_COLOR\\_FORMAT\\_I2](#). Consultare l'elenco completo nella sezione [Formati di colore](#).

#### Colori indicizzati

Per i formati di colore indicizzati (LV\_COLOR\_FORMAT\_I1/2/4/8), la palette deve essere popolata per tutti gli indici di palette che verranno utilizzati tramite [lv\\_canvas\\_set\\_palette\(canvas, index, color\)](#). Ad esempio, il seguente comando imposta i pixel con `index==3` sul rosso.

```
lv_canvas_set_palette(canvas, 3, lv_color_hex(0xffff00))
```

#### Drawing

Per impostare il colore di un singolo pixel sulla Canvas, utilizzare [lv\\_canvas\\_set\\_px\(canvas, x, y, color, opa\)](#). Con i formati di colore indicizzati (LV\_COLOR\_FORMAT\_I1/2/4/8) passare l'indice di colore come argomento `color` utilizzando il canale `blu` nel valore `color` ad esempio [lv\\_color\\_make\(0, 0, index\)](#).

[lv\\_canvas\\_fill\\_bg\(canvas, lv\\_color\\_hex\(0x00ff00\), LV\\_OPA\\_50\)](#) riempie l'intera Canvas di blu con un'opacità del 50%. Si noti che se il formato colore corrente non supporta i colori (ad esempio [LV\\_COLOR\\_FORMAT\\_A8](#)), il colore verrà ignorato. Allo stesso modo, se l'opacità non è supportata (ad esempio [LV\\_COLOR\\_FORMAT\\_RGB565](#)), verrà ignorata.

Un array di pixel può essere copiato sulla Canvas con [lv\\_canvas\\_copy\\_buf\(canvas, buffer\\_to\\_copy, x, y, width, height\)](#). Il formato colore del buffer e della Canvas devono corrispondere.

Per disegnare qualcosa sul Canvas, utilizzare direttamente le funzioni di disegno di LVGL. Vedere gli esempi per maggiori dettagli.

Le funzioni di disegno possono disegnare in qualsiasi formato colore supportato da LVGL. In genere, questo significa [LV\\_COLOR\\_FORMAT\\_RGB565](#), [LV\\_COLOR\\_FORMAT\\_RGB888](#), [LV\\_COLOR\\_FORMAT\\_XRGB888](#) e [LV\\_COLOR\\_FORMAT\\_ARGB8888](#).

### 6.10.4 Eventi

I Widget Canvas non inviano eventi speciali.

### **❶ Ulteriori Informazioni**

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

## 6.10.5 Tasti

I Widget Canvas non elaborano alcun *Tasto*.

### **❶ Ulteriori Informazioni**

Scoprire di più sui *Tasti*.

## 6.10.6 Esempio

### Disegno sulla Canvas e rotazione

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 200
#define CANVAS_HEIGHT 150

void lv_example_canvas_1(void)
{
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.radius = 10;
    rect_dsc.bg_opa = LV_OPA_COVER;
    rect_dsc.bg_grad.dir = LV_GRAD_DIR_VER;
    rect_dsc.bg_grad.stops[0].color = lv_palette_main(LV_PALETTE_RED);
    rect_dsc.bg_grad.stops[0].opa = LV_OPA_100;
    rect_dsc.bg_grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
    rect_dsc.bg_grad.stops[1].opa = LV_OPA_50;
    rect_dsc.border_width = 2;
    rect_dsc.border_opa = LV_OPA_90;
    rect_dsc.border_color = lv_color_white();
    rect_dsc.shadow_width = 5;
    rect_dsc.shadow_offset_x = 5;
    rect_dsc.shadow_offset_y = 5;

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_palette_main(LV_PALETTE_ORANGE);
    label_dsc.text = "Some text on text canvas";
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf_16bpp, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_RGB565);
    LV_DRAW_BUF_INIT_STATIC(draw_buf_16bpp);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf_16bpp);
    lv_obj_center(canvas);
    lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_area_t coords_rect = {30, 20, 100, 70};
    lv_draw_rect(&layer, &rect_dsc, &coords_rect);

    lv_area_t coords_text = {40, 80, 100, 120};
    lv_draw_label(&layer, &label_dsc, &coords_text);

    lv_canvas_finish_layer(canvas, &layer);

    /*Test the rotation. It requires another buffer where the original image is stored.
     *So use previous canvas as image and rotate it to the new canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf_32bpp, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf_32bpp);

    /*Create a canvas and initialize its palette*/
    canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf_32bpp);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 1), LV_OPA_COVER);

    lv_canvas_init_layer(canvas, &layer);
    lv_image_dsc_t img;
    lv_draw_buf_to_image(&draw_buf_16bpp, &img);
    lv_draw_image_dsc_t img_dsc;
    lv_draw_image_dsc_init(&img_dsc);
    img_dsc.rotation = 120;
    img_dsc.src = &img;
    img_dsc.pivot.x = CANVAS_WIDTH / 2;
    img_dsc.pivot.y = CANVAS_HEIGHT / 2;

    lv_area_t coords_img = {0, 0, CANVAS_WIDTH - 1, CANVAS_HEIGHT - 1};
    lv_draw_image(&layer, &img_dsc, &coords_img);

    lv_canvas_finish_layer(canvas, &layer);
}
```

(continues on next page)

(continua dalla pagina precedente)

#endif

## Canvas trasparente con chroma key

```
#include "../../../lv_examples.h"
#ifndef LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 80
#define CANVAS_HEIGHT 40

/***
 * Create a transparent canvas with transparency
 */
void lv_example_canvas_2(void)
{
    lv_obj_set_style_bg_color(lv_screen_active(), lv_palette_lighten(LV_PALETTE_RED, 5), 0);

    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_obj_center(canvas);

    /*Red background (There is no dedicated alpha channel in indexed images so LV_OPA_COVER is ignored)*/
    lv_canvas_fill_bg(canvas, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_COVER);

    /*Create hole on the canvas*/
    int32_t x;
    int32_t y;
    for(y = 10; y < 20; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_50);
        }
    }

    for(y = 20; y < 30; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_20);
        }
    }

    for(y = 30; y < 40; y++) {
        for(x = 5; x < 75; x++) {
            lv_canvas_set_px(canvas, x, y, lv_palette_main(LV_PALETTE_BLUE), LV_OPA_0);
        }
    }
}
#endif
```

## Disegnare un rettangolo sul canvas

```
#include "../../../lv_examples.h"
#ifndef LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a rectangle to the canvas
 */
void lv_example_canvas_3(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);

    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_rect_dsc_t dsc;
    lv_draw_rect_dsc_init(&dsc);
    dsc.bg_color = lv_palette_main(LV_PALETTE_RED);
    dsc.border_color = lv_palette_main(LV_PALETTE_BLUE);
    dsc.border_width = 3;
    dsc.outline_color = lv_palette_main(LV_PALETTE_GREEN);
    dsc.outline_width = 2;
    dsc.outline_pad = 2;
    dsc.outline_opa = LV_OPA_50;
    dsc.radius = 5;
    dsc.border_width = 3;

    lv_area_t coords = {10, 10, 40, 30};

    lv_draw_rect(&layer, &dsc, &coords);
    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

## Disegnare un'etichetta sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_FONT_MONTserrat_18 && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a text to the canvas
 */
void lv_example_canvas_4(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_label_dsc_t dsc;
    lv_draw_label_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.font = &lv_font_montserrat_18;
    dsc.decor = LV_TEXT_DECOR_UNDERLINE;
    dsc.text = "Hello";

    lv_area_t coords = {10, 10, 30, 60};
    lv_draw_label(&layer, &dsc, &coords);
    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

## Disegnare un arco sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw an arc to the canvas
 */
void lv_example_canvas_5(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_arc_dsc_t dsc;
    lv_draw_arc_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.width = 5;
    dsc.center.x = 25;
    dsc.center.y = 25;
    dsc.width = 10;
    dsc.radius = 15;
    dsc.start_angle = 0;
    dsc.end_angle = 220;

    lv_draw_arc(&layer, &dsc);
    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

## Disegnare un'immagine sul canvas

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw an image to the canvas
 */
void lv_example_canvas_6(void)
{
    /*Create a buffer for the canvas*/
    static uint8_t cbuf[LV_CANVAS_BUF_SIZE(CANVAS_WIDTH, CANVAS_HEIGHT, 32, LV_DRAW_BUF_STRIDE_ALIGN)];

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
lv_obj_center(canvas);

lv_layer_t layer;
lv_canvas_init_layer(canvas, &layer);

LV_IMAGE_DECLARE(img_star);
lv_draw_image_dsc_t dsc;
lv_draw_image_dsc_init(&dsc);
dsc.src = &img_star;

lv_area_t coords = {10, 10, 10 + img_star.header.w - 1, 10 + img_star.header.h - 1};

lv_draw_image(&layer, &dsc, &coords);

lv_canvas_finish_layer(canvas, &layer);
}
#endif

```

## Disegnare una linea sul canvas

```

#include "../../../lv_examples.h"
#if LV_USE_CANVAS&& LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a line to the canvas
 */
void lv_example_canvas_7(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_line_dsc_t dsc;
    lv_draw_line_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.width = 4;
    dsc.round_end = 1;
    dsc.round_start = 1;
    dsc.p1.x = 15;
    dsc.p1.y = 15;
    dsc.p2.x = 35;
    dsc.p2.y = 10;
    lv_draw_line(&layer, &dsc);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif

```

## Disegnare un'immagine vettoriale sul canvas

```

#include "../../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#if LV_USE_VECTOR_GRAPHIC

#define CANVAS_WIDTH 150
#define CANVAS_HEIGHT 150

/***
 * Draw a path to the canvas
 */
void lv_example_canvas_8(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_vector_dsc_t * dsc = lv_vector_dsc_create(&layer);
    lv_vector_path_t * path = lv_vector_path_create(LV_VECTOR_PATH_QUALITY_MEDIUM);

    lv_fpoint_t pts[] = {{10, 10}, {130, 130}, {10, 130}};
    lv_vector_path_move_to(path, &pts[0]);
    lv_vector_path_line_to(path, &pts[1]);
    lv_vector_path_line_to(path, &pts[2]);
    lv_vector_path_close(path);

    lv_vector_dsc_set_fill_color(dsc, lv_color_make(0x00, 0x80, 0xff));
    lv_vector_dsc_add_path(dsc, path);

    lv_draw_vector(dsc);
    lv_vector_path_delete(path);
}
#endif

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_vector_dsc_delete(dsc);
    lv_canvas_finish_layer(canvas, &layer);
}
#else

void lv_example_canvas_8(void)
{
    /*fallback for online examples*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Vector graphics is not enabled");
    lv_obj_center(label);
}

#endif /*LV_USE_VECTOR_GRAPHIC*/
#endif

```

## Disegnare un triangolo sul canvas

```

#include "../../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 150
#define CANVAS_HEIGHT 150

/***
 * Draw a triangle to the canvas
 */
void lv_example_canvas_9(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_triangle_dsc_t tri_dsc;
    lv_draw_triangle_dsc_init(&tri_dsc);
    tri_dsc.p[0].x = 10;
    tri_dsc.p[0].y = 10;
    tri_dsc.p[1].x = 100;
    tri_dsc.p[1].y = 30;
    tri_dsc.p[2].x = 50;
    tri_dsc.p[2].y = 100;

    tri_dsc.grad.stops_count = 2;
    tri_dsc.grad.dir = LV_GRAD_DIR_VER;
    tri_dsc.grad.stops[0].color = lv_color_hex(0xff0000);
    tri_dsc.grad.stops[0].frac = 64; /*Start at 25%*/
    tri_dsc.grad.stops[0].opa = LV_OPA_COVER;
    tri_dsc.grad.stops[1].color = lv_color_hex(0x0000ff);
    tri_dsc.grad.stops[1].opa = LV_OPA_TRANS;
    tri_dsc.grad.stops[1].frac = 3 * 64; /*End at 75%*/

    tri_dsc.opa = 128; /*Set the overall opacity to 50%*/
    lv_draw_triangle(&layer, &tri_dsc);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif

```

## Disegnare Effetti di Lettere Fantasiose

```

#include "../../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 300
#define CANVAS_HEIGHT 200

static void timer_cb(lv_timer_t * timer)
{
    static int16_t counter = 0;
    const char * string = "lol~ I'm waaaaaaaaing>>>";
    const int16_t string_len = lv_strlen(string);

    lv_obj_t * canvas = (lv_obj_t *)lv_timer_get_user_data(timer);
    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_COVER);

    lv_draw_letter_dsc_t letter_dsc;
    lv_draw_letter_dsc_init(&letter_dsc);
    letter_dsc.color = lv_color_hex(0xffff00);
    letter_dsc.font = lv_font_get_default();

    {
#define CURVE2_X(t) (t * 2 + 10)
#define CURVE2_Y(t) (lv_trigo_sin((t) * 5) * 40 / 32767 + CANVAS_HEIGHT / 2)

        int32_t pre_x = CURVE2_X(-1);
        int32_t pre_y = CURVE2_Y(-1);
        for(int16_t i = 0; i < string_len; i++) {

```

(continues on next page)

(continua dalla pagina precedente)

```

const int16_t angle = (int16_t)(i * 5);
const int32_t x = CURVE2_X(angle);
const int32_t y = CURVE2_Y(angle + counter / 2);
const lv_point_t point = { .x = x, .y = y };

letter_dsc.unicode = (uint32_t)string[i % string_len];
letter_dsc.rotation = lv_atan2(y - pre_y, x - pre_x) * 10;
letter_dsc.color = lv_color_hsv_to_rgb(i * 10, 100, 100);
lv_draw_letter(&layer, &letter_dsc, &point);

pre_x = x;
pre_y = y;
}

lv_canvas_finish_layer(canvas, &layer);
counter++;
}

void lv_example_canvas_10(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_obj_set_size(canvas, CANVAS_WIDTH, CANVAS_HEIGHT);

    lv_obj_center(canvas);

    lv_canvas_set_draw_buf(canvas, &draw_buf);

    lv_timer_create(timer_cb, 16, canvas);
}
#endif

```

## Disegnare Effetti di Lettere Fantasiose 2

```

#include "../../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 300
#define CANVAS_HEIGHT 200

static void timer_cb(lv_timer_t * timer)
{
    static int32_t counter = 0;
    const char * string = "windstorrrrrrrrrrrrrm~>>";
    const int16_t string_len = lv_strlen(string);

    lv_obj_t * canvas = (lv_obj_t *) lv_timer_get_user_data(timer);
    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_COVER);

    lv_draw_letter_dsc_t letter_dsc;
    lv_draw_letter_dsc_init(&letter_dsc);
    letter_dsc.color = lv_color_hex(0xffff0000);
    letter_dsc.font = lv_font_get_default();

    {
#define CURVE2_X(t) (((t) * 2 + lv_trigo_cos((t) * 5) * 40 / 32767 - 10)
#define CURVE2_Y(t, T) ((t) * lv_trigo_sin(((t) + (T)) * 5) * 40 / 32767 / 80 + CANVAS_HEIGHT / 2)

        int32_t pre_x = CURVE2_X(-1);
        int32_t pre_y = CURVE2_Y(-1, 0);
        for(int16_t i = 0; i < string_len; i++) {
            const int32_t angle = i * 5;
            const int32_t x = CURVE2_X(angle);
            const int32_t y = CURVE2_Y(angle + 30, counter / 2);

            letter_dsc.unicode = (uint32_t)string[i % string_len];
            letter_dsc.rotation = lv_atan2(y - pre_y, x - pre_x) * 10;
            letter_dsc.color = lv_color_hsv_to_rgb(i * 10, 100, 100);
            lv_point_t p = (lv_point_t) {
                .x = x, .y = y
            };
            lv_draw_letter(&layer, &letter_dsc, &p);

            pre_x = x;
            pre_y = y;
        }
    }

    lv_canvas_finish_layer(canvas, &layer);
    counter++;
}

void lv_example_canvas_11(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_obj_set_size(canvas, CANVAS_WIDTH, CANVAS_HEIGHT);

    lv_obj_center(canvas);

    lv_canvas_set_draw_buf(canvas, &draw_buf);

    lv_timer_create(timer_cb, 16, canvas);
}
#endif

```

## 6.10.7 API

*lv\_canvas.h*

*lv\_canvas\_private.h*

## 6.11 Chart (lv\_chart)

### 6.11.1 Panoramica

I Widget "Chart" [grafici] vengono utilizzati per visualizzare i dati.

I Chart hanno:

- 0 o più linee di divisione di sfondo (orizzontali e verticali),
- 4 assi interni, ognuno dei quali può essere utilizzato per specificare la scala per una serie di dati,
- un `point_count` (10 per default) che si applica a tutte le serie di dati aggiunte,
- 0 o più serie di dati (è possibile aggiungerle o rimuoverle in qualsiasi momento),
- 0 o più cursori (è possibile aggiungerli o rimuoverli in qualsiasi momento),
- modalità di aggiornamento (modifica il comportamento di `lv_chart_set_next_value()` se utilizzato).

I Chart possono:

- visualizzare assi flessibili con tacche e testo
- mostrare posizioni precise di punti o altre posizioni sul grafico con i cursori
- mostrare o nascondere singole serie di dati
- mostrare o nascondere singoli punti dati
- mostrare o nascondere i cursori
- eseguire scroll e zoom

### 6.11.2 Nozioni di Base sui Chart

Ogni chart ha i seguenti attributi (oltre agli attributi presenti in *all Widgets*):

Tipo (regola il modo in cui vengono disegnate le serie di dati di un grafico)

- Can be LINE (default), BAR, STACKED, SCATTER, or NONE.
- È possibile modificare il tipo di grafico in qualsiasi momento durante la sua implementazione.

Linee di Divisione Orizzontali e Verticali

- rispettivamente 3 e 5 di default
- può essere qualsiasi valore non negativo, incluso 0

2 assi Y e 2 assi X (questi ultimi vengono utilizzati con i grafici SCATTER [a dispersione])

- Tutti e 4 gli assi sono inclusi automaticamente in ogni grafico (non è necessario crearli).
- I loro intervalli predefiniti sono [0..100]. Se è necessario un intervallo diverso, impostarlo prima di disegnare il grafico.
- Un asse si "utilizza" associandolo a una serie di dati, cosa che avviene al momento della creazione della serie di dati (maggiori dettagli di seguito). È possibile associare più serie di dati a ciascun asse.

Conteggio punti (numero di punti dati in tutte le serie di dati aggiunte al grafico)

- il default è 10

- Se si forniscono array di valori dati personalizzati, ogni array fornito deve contenere almeno questo numero di valori.
- For LINE-, BAR-, STACKED-charts, this is the number of points on the X axis.
- LINE-, BAR-, STACKED-charts require only one data-value array to supply Y-values for each data point.
- Per i grafici a dispersione, questo è il numero di punti a dispersione nella serie di dati.
- I grafici a dispersione hanno array di valori dati separati sia per i valori X che per i valori Y.

Qualsiasi numero di serie di dati

- Dopo la creazione, un grafico inizialmente non contiene alcuna serie di dati. Si devono aggiungere.
- È possibile aggiungere e rimuovere serie di dati in qualsiasi momento durante la vita di un grafico.
- Quando viene creata una serie di dati, questa viene fornita con array di valori pre-allocati in base al tipo di grafico e al **point\_count**. (Tutti i tipi di grafico utilizzano un array di valori Y. Anche i grafici di tipo SCATTER utilizzano un array di valori X). Tutti i valori Y così allocati vengono impostati su **LV\_CHART\_POINT\_NONE**, il che fa sì che il punto venga nascosto.
- Per far sì che i punti vengano disegnati sul grafico, è necessario impostare i loro valori Y su un valore diverso da **LV\_CHART\_POINT\_NONE**.
- È possibile nascondere un punto impostando il suo valore Y su **LV\_CHART\_POINT\_NONE**.
- Se lo si desidera, è possibile indicare a una serie di dati di utilizzare invece un array di valori fornito. In tal caso:
  - Gli array di valori pre-allocati vengono automaticamente liberati.
  - Quella serie di dati continuerà a utilizzare *l'array fornito* da quel momento in poi.
  - I valori nell'array devono rimanere disponibili per tutta la durata di vita di quella serie di dati.
  - È necessario assicurarsi che ogni array fornito contenga almeno **point\_count int32\_t** elementi.
  - La gestione della durata di vita di tutti gli array di valori forniti è a discrezione dell'utente.

Un numero qualsiasi di cursori

- Dopo la creazione di un grafico, inizialmente non contiene cursori. È necessario aggiungerli se si desidera utilizzarli.
- È possibile aggiungere, mostrare, nascondere o rimuovere i cursori in qualsiasi momento durante la vita di un grafico.

Modalità di aggiornamento

- *Vedere di seguito*

## Layer di Chart

Quando si disegna un grafico, alcuni elementi appaiono sopra altri, in questo ordine, dal basso verso l'alto:

- Lo sfondo del grafico (con linee di divisione opzionali)
- Ogni serie di dati:
  - Le prime serie di dati aggiunte appaiono in alto.
  - Per un grafico a dispersione, all'interno di ogni serie, i punti successivi nella sequenza appariranno sopra i punti precedenti nella sequenza in caso di sovrapposizione.
- Ogni cursore (se presente):
  - Il cursore aggiunto più di recente appare in alto.

### 6.11.3 Parti e Stili

- ***LV\_PART\_MAIN*** The background of the chart. Uses the *typical background* and line style properties (for division lines). *Padding* makes the series area smaller. For BAR and STACKED charts **pad\_column** sets the space between bars in the same data series.
- ***LV\_PART\_SCROLLBAR*** Una scrollbar utilizzata se il grafico è ingrandito. Vedere la documentazione del *Widget Base* (*lv\_obj*).
- ***LV\_PART\_ITEMS*** Si riferisce alle serie di dati LINE o BAR.
  - Grafico a LINEE: le proprietà *line* sono utilizzate dalle linee. **width**, **height**, **bg\_color** e **radius** vengono utilizzate per impostare l'aspetto dei punti sulla linea.
  - Grafico a barre: le tipiche proprietà di sfondo vengono utilizzate per definire lo stile delle barre. **pad\_column** impone lo spazio tra le colonne nella stessa serie di dati.
- ***LV\_PART\_INDICATOR*** Si riferisce ai punti sui grafici a LINEE e SCATTER (piccoli cerchi o quadrati [con angoli eventualmente arrotondati]).
- ***LV\_PART\_CURSOR*** Le proprietà *Line* vengono utilizzate per definire lo stile dei cursori. **width**, **height**, **bg\_color** e **radius** vengono utilizzati per impostare l'aspetto del "punto" del cursore, indicandone la posizione. Se **width** o **height** sono impostati su 0, vengono disegnate solo le linee del cursore.

### 6.11.4 Dettagli

#### Tipo di Chart

Un "chart" [grafico] può essere di uno dei seguenti tipi:

- ***LV\_CHART\_TYPE\_NONE***: Non visualizza alcun dato. Può essere utilizzato per nascondere i dati del grafico.
- ***LV\_CHART\_TYPE\_LINE***: Disegna linee tra i punti dati. I punti dati possono anche essere visualizzati se i relativi stili **width**, **height**, **bg\_color** e **radius** (per ***LV\_PART\_ITEMS***) sono impostati e sia **width** che **height** hanno valori diversi da zero.
- ***LV\_CHART\_TYPE\_BAR***: Draw individual vertical bars for each point in each series.
- ***LV\_CHART\_TYPE\_STACKED***: Draw vertical stacked bars where multiple data series are displayed as segments within a single bar for each data point. Supports only positive values.
- ***LV\_CHART\_TYPE\_SCATTER***: il grafico X/Y traccia i punti e, facoltativamente, le linee tra i punti se i valori dello stile di larghezza della linea per ***LV\_PART\_ITEMS*** sono diversi da zero e il valore Y del punto è diverso da ***LV\_CHART\_POINT\_NONE***. (Il disegno di singoli punti su un grafico a dispersione può essere soppresso se i loro valori Y sono impostati su ***LV\_CHART\_POINT\_NONE***).

I grafici iniziano la loro vita come grafici a linee. È possibile modificare il tipo di un grafico con ***lv\_chart\_set\_type(chart, LV\_CHART\_TYPE\_...)***.

#### Serie di dati

È possibile aggiungere un numero qualsiasi di serie di dati a un grafico utilizzando

```
lv_chart_add_series(chart, color, axis).
```

Questo alloca (e restituisce un puntatore a) una struttura **lv\_chart\_series\_t** che memorizza **color** e **axis** specificati e viene preallocata con un array di valori Y **chart->point\_cnt int32\_t**, tutti impostati su ***LV\_CHART\_POINT\_NONE***. (Un grafico a dispersione include anche un array preassegnato dello stesso numero di valori X).

**axis** specifica quale asse viene utilizzato per scalare i valori e può essere uno dei seguenti:

- ***LV\_CHART\_AXIS\_PRIMARY\_Y***: Asse sinistro
- ***LV\_CHART\_AXIS\_SECONDARY\_Y***: Asse destro
- ***LV\_CHART\_AXIS\_PRIMARY\_X***: Asse inferiore

- `LV_CHART_AXIS_SECONDARY_X`: Asse superiore

Quando si aggiunge una serie di dati a un grafico a dispersione, è possibile utilizzare l'opzione "bit per bit" OPPURE l'asse Y selezionato (primario o secondario) con uno dei valori dell'asse X.

Se si desidera che il grafico utilizzi il proprio array di valori Y invece di quello fornito, lo si può fare con

`lv_chart_set_series_ext_y_array(chart, series, value_array)`.

Si è responsabili di garantire che l'array fornito contenga almeno `chart->point_cnt` elementi.

`value_array` dovrebbe apparire così: `int32_t * value_array[num_points]`. Solo il puntatore dell'array viene salvato nella serie, quindi il suo contenuto deve rimanere disponibile per tutta la durata della serie, ovvero l'array deve essere globale, statico o allocato dinamicamente.

#### Nota

Chiamare `lv_chart_refresh(chart)` quando i dati di un grafico sono cambiati per segnalare che il grafico deve essere nuovamente visualizzato al successivo aggiornamento del display. Non è necessario farlo se si utilizzano gli array di valori forniti e si impostano i valori con le funzioni `lv_chart_set_...value_...`(). Vedere di seguito per ulteriori informazioni su queste funzioni.

Un puntatore all'array di valori Y di una serie può essere ottenuto con `lv_chart_get_series_y_array(chart, series)`. Questo vale sia che si utilizzi l'array di valori Y fornito sia che se ne fornisca uno proprio.

Per i grafici di tipo SCATTER,

- Anche `lv_chart_set_series_ext_x_array(chart, series, value_array)` e
- `lv_chart_get_series_x_array(chart, series)`

Possono essere utilizzati.

### Modifica dei dati

Sono disponibili diverse opzioni per impostare i valori Y per una serie di dati:

1. Impostare i valori a livello di codice nell'array, ad esempio `ser1->points[3] = 7` e aggiornare il grafico con `lv_chart_refresh(chart)`.
2. Utilizzare `lv_chart_set_series_value_by_id(chart, series, id, value)` dove `id` è l'indice a partire da zero del punto che si desidera aggiornare.
3. Usare `lv_chart_set_next_value(chart, series, value)`. (Vedere le *Modalità di aggiornamento* di seguito.)
4. Impostare tutti i punti su un singolo valore Y con `lv_chart_set_all_values(chart, series, value)`.

Usare `LV_CHART_POINT_NONE` come valore per fare in modo che la libreria salti il disegno di quel punto, colonna o punto di dispersione.

Per i grafici di tipo SCATTER,

- `lv_chart_set_series_value_by_id2(chart, series, id, x_value, y_value)` e
- `lv_chart_set_next_value2(chart, series, x_value, y_value)`

Possono essere utilizzati.

### Modalità di aggiornamento

`lv_chart_set_next_value()` può comportarsi in due modi a seconda della *modalità di aggiornamento*:

- `LV_CHART_UPDATE_MODE_SHIFT`: Sposta i vecchi dati a sinistra e aggiunge quelli nuovi a destra.

- `LV_CHART_UPDATE_MODE_CIRCULAR`: Aggiunge i nuovi dati in modo circolare, come un diagramma ECG.

La modalità di aggiornamento può essere modificata con `lv_chart_set_update_mode(chart, LV_CHART_UPDATE_MODE_...)`.

### Numero di punti

Il numero di punti nella serie può essere modificato con `lv_chart_set_point_count(chart, point_num)`. Il valore di default è 10. Nota: questo influisce sul numero di punti elaborati quando un array di valori esterni viene assegnato a una serie, quindi è necessario assicurarsi che qualsiasi array esterno fornito contenga almeno `point_num` elementi.

### Gestione di un numero elevato di punti

Nei grafici LINE, se il numero di punti è maggiore dei pixel in orizzontale, il grafico disegnerà solo linee verticali per rendere efficace la visualizzazione di grandi quantità di dati. Se, ad esempio, ci sono 10 punti per pixel, LVGL cerca il valore più piccolo e quello più grande e disegna delle linee verticali tra di essi per garantire che non vengano trascurati picchi.

### Range verticale

È possibile specificare i valori minimo e massimo in direzione Y con `lv_chart_set_axis_range(chart, axis, min, max)`. `axis` può essere `LV_CHART_AXIS_PRIMARY_Y` (asse Y sinistro) o `LV_CHART_AXIS_SECONDARY_Y` (asse Y destro).

Il valore dei punti verrà scalato proporzionalmente. L'intervallo di default è 0..100.

### Linee di divisione

Il numero di linee di divisione orizzontali e verticali può essere modificato tramite `lv_chart_set_div_line_count(chart, hdiv_num, vdiv_num)`. Le impostazioni di default sono 3 linee di divisione orizzontali e 5 verticali. Se è presente un bordo visibile su un lato e non è presente alcun "padding" [riempimento], la linea di divisione verrà disegnata sopra il bordo e in questo caso non verrà disegnata per non nascondere il bordo del grafico.

### Ignorare il punto di inizio di default per le serie

Per far sì che un grafico inizi da un punto diverso da quello predefinito, ovvero `point[0]` della serie, è possibile impostare un indice alternativo con la funzione `lv_chart_set_x_start_point(chart, series, id)` dove `id` la nuova posizione dell'indice a base zero da cui iniziare il grafico.

Si noti che `LV_CHART_UPDATE_MODE_SHIFT` modifica anche `start_point`.

### Segni di graduazione ed etichette

Con l'aiuto di `Scale`, è possibile aggiungere scale verticali e orizzontali in modo molto flessibile. Vedere l'[esempio 2](#) qui sotto per saperne di più.

### Zoom

Per ingrandire il grafico, è sufficiente inserirlo in un contenitore padre e impostare la larghezza e/o l'altezza del grafico su un valore maggiore. In questo modo, il grafico sarà "scrollable" [scorrevole] nel suo contenitore padre, ovvero il contenitore padre fornisce la "finestra di visualizzazione" scorrevole.

### Cursore

Inizialmente, a un nuovo cursore viene assegnata la posizione `LV_CHART_POINT_NONE`, che lo nasconde. Per visualizzare il cursore, la sua posizione deve essere impostata da programma utilizzando una delle funzioni seguenti.

È possibile nascondere un cursore senza rimuoverlo dal grafico utilizzando `lv_chart_set_cursor_point()` passando `LV_CHART_POINT_NONE` come ID del punto.

È possibile aggiungere un cursore con `lv_chart_cursor_t * c1 = lv_chart_add_cursor(chart, color, dir);`. I possibili valori di `dir` sono i valori di enumerazione di `lv_dir_t`: `LV_DIR_NONE/RIGHT/UP/LEFT/DOWN/HOR/VER/ALL` o i loro valori in OR bit a bit per indicare al grafico in quale direzione tracciare le linee.

`lv_chart_set_cursor_pos(chart, cursor, &point)` imposta la posizione del cursore su un punto arbitrario del grafico. `&point` è un puntatore a una variabile `lv_point_t`. Ad esempio, `lv_point_t point = {10, 20}`. Se si scorre il grafico, il cursore si sposta con esso.

`lv_chart_get_point_pos_by_id(chart, series, id, &point_out)` ottiene le coordinate di un dato punto del grafico. Questo è utile per posizionare il cursore su quel punto dati.

`lv_chart_set_cursor_point(chart, cursor, series, point_id)` posiziona il cursore sul punto dati specificato sul grafico. Se la posizione del punto cambia (tramite un nuovo valore o tramite scorrimento), il cursore si sposta con il punto. Vedere un esempio di utilizzo di questa funzione [qui](#).

## 6.11.5 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando viene premuto un nuovo punto sul grafico. `lv_chart_get_pressed_point(chart)` restituisce l'indice a partire da zero del punto premuto.

### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

## 6.11.6 Tasti

Nessun *tasto* viene elaborato dai Widget Chart.

### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

## 6.11.7 Esempio

### Grafico a linee

```
#include "../../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

/***
 * A very basic line chart
 */
void lv_example_chart_1(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_center(chart);
    lv_chart_set_type(chart, LV_CHART_TYPE_LINE); /*Show lines and points too*/

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_GREEN), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_SECONDARY_Y);
    int32_t * ser2_y_points = lv_chart_get_series_y_array(chart, ser2);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        /*Set the next points on 'ser1'*/
        lv_chart_set_next_value(chart, ser1, (int32_t)lv_rand(10, 50));

        /*Directly set points on 'ser2'*/
        ser2_y_points[i] = (int32_t)lv_rand(50, 90);
    }
    lv_chart_refresh(chart); /*Required after direct set*/
}

#endif
```

## Tick degli assi ed etichette con scrolling

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

/***
 * Use lv_scale to add ticks to a scrollable chart
 */
void lv_example_chart_2(void)
{
    /*Create a container*/
    lv_obj_t * main_cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(main_cont, 200, 150);
    lv_obj_center(main_cont);

    /*Create a transparent wrapper for the chart and the scale.
     *Set a large width, to make it scrollable on the main container*/
    lv_obj_t * wrapper = lv_obj_create(main_cont);
    lv_obj_remove_style_all(wrapper);
    lv_obj_set_size(wrapper, lv_pct(300), lv_pct(100));
    lv_obj_set_flex_flow(wrapper, LV_FLEX_FLOW_COLUMN);

    /*Create a chart on the wrapper
     *Set it's width to 100% to fill the large wrapper*/
    lv_obj_t * chart = lv_chart_create(wrapper);
    lv_obj_set_width(chart, lv_pct(100));
    lv_obj_set_flex_grow(chart, 1);
    lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 100);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_SECONDARY_Y, 0, 400);
    lv_chart_set_point_count(chart, 12);
    lv_obj_set_style_radius(chart, 0, 0);

    /*Create a scale also with 100% width*/
    lv_obj_t * scale_bottom = lv_scale_create(wrapper);
    lv_scale_set_mode(scale_bottom, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_set_size(scale_bottom, lv_pct(100), 25);
    lv_scale_set_total_tick_count(scale_bottom, 12);
    lv_scale_set_major_tick_every(scale_bottom, 1);
    lv_obj_set_style_pad_hor(scale_bottom, lv_chart_get_first_point_center_offset(chart), 0);

    static const char * month[] = {"Jan", "Febr", "March", "Apr", "May", "Jun", "July", "Aug", "Sept", "Oct", "Nov", "Dec", NULL};
    lv_scale_set_text_src(scale_bottom, month);

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_lighten(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_darken(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);

    /*Set the next points on 'ser1'*/
    uint32_t i;
    for(i = 0; i < 12; i++) {
        lv_chart_set_next_value(chart, ser1, (int32_t)lv_rand(10, 60));
        lv_chart_set_next_value(chart, ser2, (int32_t)lv_rand(50, 90));
    }
    lv_chart_refresh(chart); /*Required after direct set*/
}

#endif
```

## Mostrare il valore dei punti premuti

```
#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * chart = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        lv_obj_invalidate(chart);
    }
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        int32_t * s = (int32_t *)lv_event_get_param(e);
        *s = LV_MAX(*s, 20);
    }
    else if(code == LV_EVENT_DRAW_POST_END) {
        uint32_t id = lv_chart_get_pressed_point(chart);
        if(id == LV_CHART_POINT_NONE) return;

        LV_LOG_USER("Selected point %d", (int)id);

        lv_chart_series_t * ser = lv_chart_get_series_next(chart, NULL);
        int32_t value = 0;
        while(ser) {
            lv_point_t p;
            lv_chart_get_point_pos_by_id(chart, ser, id, &p);

            int32_t * y_array = lv_chart_get_series_y_array(chart, ser);
            if(y_array[id] != LV_CHART_POINT_NONE && y_array[id] >= 0) {
                /*Accumulate the values to show the rectangles at the top of each segment*/
                value += y_array[id];
            }

            /*Draw a rectangle above the clicked point*/
            lv_layer_t * layer = lv_event_get_layer(e);
            lv_draw_rect_dsc_t draw_rect_dsc;
            lv_draw_rect_dsc_init(&draw_rect_dsc);
            draw_rect_dsc.bg_color = lv_color_black();
            draw_rect_dsc.bg_opa = LV_OPA_50;
            draw_rect_dsc.radius = 3;

            lv_area_t chart_obj_coords;
            lv_obj_get_coords(chart, &chart_obj_coords);
            lv_area_t rect_area;
            rect_area.x1 = chart_obj_coords.x1 + p.x - 20;
            rect_area.x2 = chart_obj_coords.x1 + p.x + 20;
            rect_area.y1 = chart_obj_coords.y1 + p.y - 10;
            rect_area.y2 = chart_obj_coords.y1 + p.y + 10;
        }
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_draw_rect(layer, &draw_rect_dsc, &rect_area);

    /*Draw the value as label to the center of the rectangle*/
    char buf[16];
    lv_snprintf(buf, sizeof(buf), LV_SYMBOL_DUMMY"%d", value);

    lv_draw_label_dsc_t draw_label_dsc;
    lv_draw_label_dsc_init(&draw_label_dsc);
    draw_label_dsc.color = lv_color_white();
    draw_label_dsc.text = buf;
    draw_label_dsc.text_local = 1;
    draw_label_dsc.align = LV_TEXT_ALIGN_CENTER;
    lv_area_t label_area;
    lv_area_set_height(&label_area, lv_font_get_line_height(draw_label_dsc.font));
    lv_area_align(&rect_area, &label_area, LV_ALIGN_CENTER, 0, 0);
    lv_draw_label(layer, &draw_label_dsc, &label_area);
}

ser = lv_chart_get_series_next(chart, ser);
}
else if(code == LV_EVENT_RELEASED) {
    lv_obj_invalidate(chart);
}
}

/**
 * Show the value of the pressed points
 */
void lv_example_chart_3(void)
{
    /*Create a chart*/
    lv_obj_t * chart;
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 280, 180);
    lv_obj_set_style_pad_column(chart, 4, 0);
    lv_obj_center(chart);
    lv_chart_set_type(chart, LV_CHART_TYPE_STACKED);

    lv_obj_add_event_cb(chart, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    /*Add two data series*/
    lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_GREEN), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_series_t * ser3 = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_BLUE), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser1, lv_rand(15, 30));
        lv_chart_set_next_value(chart, ser2, lv_rand(15, 30));
        lv_chart_set_next_value(chart, ser3, lv_rand(15, 30));
    }
}
#endif

```

## Ricolorare le barre in base al loro valore

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = *lv_draw_dsc_base_t *lv_draw_task_get_dsc(draw_task);

    if(base_dsc->part != LV_PART_ITEMS) {
        return;
    }

    lv_draw_fill_dsc_t * fill_dsc = lv_draw_task_get_fill_dsc(draw_task);
    if(fill_dsc) {
        lv_obj_t * chart = lv_event_get_target_obj(e);
        int32_t * y_array = lv_chart_get_series_y_array(chart, lv_chart_get_series_next(chart, NULL));
        int32_t v = y_array[base_dsc->d2];

        uint8_t ratio = (uint8_t)(v * 255 / 100);
        fill_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_GREEN), lv_palette_main(LV_PALETTE_RED), ratio);
    }
}

/**
 * Recolor the bars of a chart based on their value
 */
void lv_example_chart_4(void)
{
    /*Create a chart1*/
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
    lv_chart_set_point_count(chart, 24);
    lv_obj_set_style_pad_column(chart, 2, 0);
    lv_obj_set_size(chart, 260, 160);
    lv_obj_center(chart);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_color_hex(0xffff00), LV_CHART_AXIS_PRIMARY_Y);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(chart, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    uint32_t i;
    for(i = 0; i < 24; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
    }
}
#endif

```

## Grafico a linee sfumate con linee di divisione personalizzate

```
#include "../../lv_examples.h"

#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES
#include "../../lvgl_private.h"

static void hook_division_lines(lv_event_t * e);
static void add_faded_area(lv_event_t * e);
static void draw_event_cb(lv_event_t * e);

/***
 * Add a faded area effect to the line chart and make some division lines ticker
 */
void lv_example_chart_5(void)
{
    /*Create a chart*/
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_chart_set_type(chart, LV_CHART_TYPE_LINE); /*Show lines and points too*/
    lv_obj_set_size(chart, 200, 150);
    lv_obj_set_style_pad_all(chart, 0, 0);
    lv_obj_set_style_radius(chart, 0, 0);
    lv_obj_center(chart);

    lv_chart_set_div_line_count(chart, 5, 7);

    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(chart, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 80));
    }
}

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);

    if(base_dsc->part == LV_PART_ITEMS && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_LINE) {
        add_faded_area(e);
    }

    /*Hook the division lines too*/
    if(base_dsc->part == LV_PART_MAIN && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_LINE) {
        hook_division_lines(e);
    }
}

static void add_faded_area(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_area_t coords;
    lv_obj_get_coords(obj, &coords);

    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);

    const lv_chart_series_t * ser = lv_chart_get_series_next(obj, NULL);
    lv_color_t ser_color = lv_chart_get_series_color(obj, ser);

    /*Draw a triangle below the line with some opacity gradient*/
    lv_draw_line_dsc_t * draw_line_dsc = (lv_draw_line_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_triangle_dsc_t tri_dsc;

    lv_draw_triangle_dsc_init(&tri_dsc);
    tri_dsc.p[0].x = draw_line_dsc->p1.x;
    tri_dsc.p[0].y = draw_line_dsc->p1.y;
    tri_dsc.p[1].x = draw_line_dsc->p2.x;
    tri_dsc.p[1].y = draw_line_dsc->p2.y;
    tri_dsc.p[2].x = draw_line_dsc->p1.y < draw_line_dsc->p2.y ? draw_line_dsc->p1.x : draw_line_dsc->p2.x;
    tri_dsc.p[2].y = LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y);
    tri_dsc.grad.dir = LV_GRAD_DIR_VER;

    int32_t full_h = lv_obj_get_height(obj);
    int32_t fract_uppter = (int32_t)(LV_MIN(draw_line_dsc->p1.y, draw_line_dsc->p2.y) - coords.y1) * 255 / full_h;
    int32_t fract_lower = (int32_t)(LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y) - coords.y1) * 255 / full_h;
    tri_dsc.grad.stops[0].color = ser_color;
    tri_dsc.grad.stops[0].opa = (lv_opa_t)(255 - fract_uppter);
    tri_dsc.grad.stops[0].frac = 0;
    tri_dsc.grad.stops[1].color = ser_color;
    tri_dsc.grad.stops[1].opa = (lv_opa_t)(255 - fract_lower);
    tri_dsc.grad.stops[1].frac = 255;

    lv_draw_triangle(base_dsc->layer, &tri_dsc);

    /*Draw rectangle below the triangle*/
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.bg_grad.dir = LV_GRAD_DIR_VER;
    rect_dsc.bg_grad.stops[0].color = ser_color;
    rect_dsc.bg_grad.stops[0].frac = 0;
    rect_dsc.bg_grad.stops[0].opa = (lv_opa_t)(255 - fract_lower);
    rect_dsc.bg_grad.stops[1].color = ser_color;
    rect_dsc.bg_grad.stops[1].frac = 255;
    rect_dsc.bg_grad.stops[1].opa = 0;

    lv_area_t rect_area;
    rect_area.x1 = (int32_t)draw_line_dsc->p1.x;
    rect_area.x2 = (int32_t)draw_line_dsc->p2.x - 1;
    rect_area.y1 = (int32_t)LV_MAX(draw_line_dsc->p1.y, draw_line_dsc->p2.y) - 1;
    rect_area.y2 = (int32_t)coords.y2;
    lv_draw_rect(base_dsc->layer, &rect_dsc, &rect_area);
}

static void hook_division_lines(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_draw_line_dsc_t * line_dsc = (lv_draw_line_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);

/*Vertical line*/
if(line_dsc->p1.x == line_dsc->p2.x) {
    line_dsc->color = lv_palette_lighten(LV_PALETTE_GREY, 1);
    if(base_dsc->idl == 3) {
        line_dsc->width = 2;
        line_dsc->dash_gap = 0;
        line_dsc->dash_width = 0;
    }
    else {
        line_dsc->width = 1;
        line_dsc->dash_gap = 6;
        line_dsc->dash_width = 6;
    }
}
/*Horizontal line*/
else {
    if(base_dsc->idl == 2) {
        line_dsc->width = 2;
        line_dsc->dash_gap = 0;
        line_dsc->dash_width = 0;
    }
    else {
        line_dsc->width = 2;
        line_dsc->dash_gap = 6;
        line_dsc->dash_width = 6;
    }

    if(base_dsc->idl == 1 || base_dsc->idl == 3) {
        line_dsc->color = lv_palette_main(LV_PALETTE_GREEN);
    }
    else {
        line_dsc->color = lv_palette_lighten(LV_PALETTE_GREY, 1);
    }
}
#endif

```

## Mostrare il cursore sul punto cliccato

```

#include "../../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static lv_obj_t * chart;
static lv_chart_series_t * ser;
static lv_chart_cursor_t * cursor;

static void value_changed_event_cb(lv_event_t * e)
{
    uint32_t last_id;
    lv_obj_t * obj = lv_event_get_target_obj(e);

    last_id = lv_chart_get_pressed_point(obj);
    if(last_id != LV_CHART_POINT_NONE) {
        lv_chart_set_cursor_point(obj, cursor, NULL, last_id);
    }
}

/**
 * Show cursor on the clicked point
 */
void lv_example_chart_6(void)
{
    chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, -10);

    // lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_Y, 10, 5, 6, 5, true, 40);
    // lv_chart_set_axis_tick(chart, LV_CHART_AXIS_PRIMARY_X, 10, 5, 10, 1, true, 30);

    lv_obj_add_event_cb(chart, value_changed_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_refresh_ext_draw_size(chart);

    cursor = lv_chart_add_cursor(chart, lv_palette_main(LV_PALETTE_BLUE), (lv_dir_t)(LV_DIR_LEFT | LV_DIR_BOTTOM));

    ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
    }

    // lv_chart_set_scale_x(chart, 500);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Click on a point");
    lv_obj_align_to(label, chart, LV_ALIGN_OUT_TOP_MID, 0, -5);
}

#endif

```

## Grafico a dispersione

```

#include "../../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        lv_obj_t * obj = lv_event_get_target_obj(e);
        lv_chart_series_t * ser = lv_chart_get_series_next(obj, NULL);
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_draw_rect_dsc_t * rect_draw_dsc = (lv_draw_rect_dsc_t *)lv_draw_task_get_draw_dsc(draw_task);
uint32_t cnt = lv_chart_get_point_count(obj);

/*Make older value more transparent/
rect_draw_dsc->bg_opa = (lv_opa_t)((LV_OPA_COVER * base_dsc->id2) / (cnt - 1));

/*Make smaller values blue, higher values red*/
int32_t * x_array = lv_chart_get_series_x_array(obj, ser);
int32_t * y_array = lv_chart_get_series_y_array(obj, ser);
/*dsc->id is the tells drawing order, but we need the ID of the point being drawn.*/
uint32_t start_point = lv_chart_get_x_start_point(obj, ser);
uint32_t p_act = (start_point + base_dsc->id2) % cnt; /*Consider start point to get the index of the array*/
lv_opa_t x_opa = (lv_opa_t)((x_array[p_act] * LV_OPA_50) / 200);
lv_opa_t y_opa = (lv_opa_t)((y_array[p_act] * LV_OPA_50) / 1000);

rect_draw_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_RED),
                                         lv_palette_main(LV_PALETTE_BLUE),
                                         x_opa + y_opa);
}

static void add_data(lv_timer_t * timer)
{
    lv_obj_t * chart = (lv_obj_t *)lv_timer_get_user_data(timer);
    lv_chart_set_next_value2(chart, lv_chart_get_series_next(chart, NULL), (int32_t)lv_rand(0, 200),
                            (int32_t)lv_rand(0, 1000));
}

/***
* A scatter chart
*/
void lv_example_chart_7(void)
{
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_obj_set_size(chart, 200, 150);
    lv_obj_align(chart, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(chart, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(chart, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_obj_set_style_line_width(chart, 0, LV_PART_ITEMS); /*Remove the lines*/

    lv_chart_set_type(chart, LV_CHART_TYPE_SCATTER);

    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_X, 0, 200);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 1000);

    lv_chart_set_point_count(chart, 50);

    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    uint32_t i;
    for(i = 0; i < 50; i++) {
        lv_chart_set_next_value2(chart, ser, (int32_t)lv_rand(0, 200), (int32_t)lv_rand(0, 1000));
    }

    lv_timer_create(add_data, 100, chart);
}
#endif

```

## Grafico a linee circolari con spazio

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES

static void add_data(lv_timer_t * t)
{
    lv_obj_t * chart = (lv_obj_t *)lv_timer_get_user_data(t);
    lv_chart_series_t * ser = lv_chart_get_series_next(chart, NULL);

    lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));

    uint32_t p = lv_chart_get_point_count(chart);
    uint32_t s = lv_chart_get_x_start_point(chart, ser);
    int32_t * a = lv_chart_get_series_y_array(chart, ser);

    a[(s + 1) % p] = LV_CHART_POINT_NONE;
    a[(s + 2) % p] = LV_CHART_POINT_NONE;
    a[(s + 2) % p] = LV_CHART_POINT_NONE;

    lv_chart_refresh(chart);
}

/***
* Circular line chart with gap
*/
void lv_example_chart_8(void)
{
    /*Create a stacked_area_chart.obj*/
    lv_obj_t * chart = lv_chart_create(lv_screen_active());
    lv_chart_set_update_mode(chart, LV_CHART_UPDATE_MODE_CIRCULAR);
    lv_obj_set_size(chart, 0, 0, LV_PART_INDICATOR);
    lv_obj_set_size(chart, 280, 150);
    lv_obj_center(chart);

    lv_chart_set_point_count(chart, 80);
    lv_chart_series_t * ser = lv_chart_add_series(chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    /*Prefill with data*/
    uint32_t i;
    for(i = 0; i < 80; i++) {
        lv_chart_set_next_value(chart, ser, (int32_t)lv_rand(10, 90));
    }

    lv_timer_create(add_data, 300, chart);
}
#endif

```

## 6.11.8 API

`lv_api_map_v9_1.h`

`lv_chart.h`

`lv_chart_private.h`

## 6.12 Checkbox (lv\_checkbox)

### 6.12.1 Panoramica

Il Widget Checkbox viene creato da una "tick box" [casella di controllo] e da un'etichetta. Quando si clicca sulla Checkbox, la casella di controllo viene attivata/disattivata.

### 6.12.2 Parti e Stili

- **LV\_PART\_MAIN** Sfondo della Checkbox e utilizza il testo e le *tipiche proprietà di stile del background*. `pad_column` regola la spaziatura tra la tickbox e l'etichetta.
- **LV\_PART\_INDICATOR** La "tick box" è un quadrato che utilizza le *tipiche proprietà di stile del background*. Per default, la sua dimensione è pari all'altezza del font della parte "main". Le proprietà di "padding" [riempimento] ingrandiscono la tick box nelle rispettive direzioni.

La Checkbox aggiunta al gruppo di default (se impostato).

### 6.12.3 Utilizzo

#### Text

Il testo può essere modificato con `lv_checkbox_set_text(cb, "New text")` e verrà allocato dinamicamente.

Per impostare un testo statico, utilizzare `lv_checkbox_set_text_static(cb, txt)`. In questo modo, verrà memorizzato solo un puntatore a `txt`. Il buffer di testo fornito deve rimanere disponibile per tutta la durata della Checkbox.

#### Selezionare, deselectare, disabilitare

È possibile il "check" [selezione], "un-check" [deselezionare] e disabilitare la Checkbox tramite codice utilizzando la funzione di aggiunta/cancellazione dello stato comune:

```
lv_obj_add_state(cb, LV_STATE_CHECKED); /* Make Checkbox checked */
lv_obj_remove_state(cb, LV_STATE_CHECKED); /* Make Checkbox unchecked */
lv_obj_add_state(cb, LV_STATE_CHECKED); /* Make Checkbox checked */
lv_obj_add_state(cb, LV_STATE_DISABLED); /* Make Checkbox disabled */
```

Per scoprire se la Checkbox è selezionata, si usa `lv_obj_has_state(cb, LV_STATE_CHECKED)`.

### 6.12.4 Eventi

- **LV\_EVENT\_VALUE\_CHANGED** Invia quando la Checkbox viene attivata/disattivata.

#### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

## 6.12.5 Tasti

I seguenti *Tasti* vengono elaborati dalla casella di controllo:

- **LV\_KEY\_RIGHT/UP** Passa allo stato CHECKED se la Checkbox è abilitata
- **LV\_KEY\_LEFT/DOWN** Passa allo stato non-CHECKED se la Checkbox è abilitata
- **LV\_KEY\_ENTER** Clicca sulla Checkbox e ne cambia il valore.

Si noti che, come di consueto, lo stato di **LV\_KEY\_ENTER** viene tradotto in **LV\_EVENT\_PRESSED/PRESSING/RELEASED** ecc.

### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

## 6.12.6 Esempio

### Checkbox Semplici

```
#include "../../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_UNUSED(obj);
        const char * txt = lv_checkbox_get_text(obj);
        const char * state = lv_obj_get_state(obj) & LV_STATE_CHECKED ? "Checked" : "Unchecked";
        LV_UNUSED(txt);
        LV_UNUSED(state);
        LV_LOG_USER("%s: %s", txt, state);
    }
}

void lv_example_checkbox_1(void)
{
    lv_obj_set_flex_flow(lv_screen_active(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * cb;
    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Apple");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Banana");
    lv_obj_add_state(cb, LV_STATE_CHECKED);
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_checkbox_set_text(cb, "Lemon");
    lv_obj_add_state(cb, LV_STATE_DISABLED);
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_screen_active());
    lv_obj_add_state(cb, LV_STATE_CHECKED);
    lv_obj_add_state(cb, LV_STATE_DISABLED);
    lv_checkbox_set_text(cb, "Melon\nand a new line");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    lv_obj_update_layout(cb);
}

#endif
```

### Checkboxes come come radio button

```
#include "../../lv_examples.h"
#if LV_USE_CHECKBOX && LV_BUILD_EXAMPLES

static lv_style_t style_radio;
static lv_style_t style_radio_chk;
static int32_t active_index_1 = 0;
static int32_t active_index_2 = 0;

static void radio_event_handler(lv_event_t * e)
{
    int32_t * active_id = (int32_t *)lv_event_get_user_data(e);
    lv_obj_t * cont = (lv_obj_t *)lv_event_get_current_target(e);
    lv_obj_t * act_cb = lv_event_get_target_obj(e);
    lv_obj_t * old_cb = lv_obj_get_child(cont, *active_id);

    /*Do nothing if the container was clicked*/
    if(act_cb == cont) return;

    lv_obj_remove_state(old_cb, LV_STATE_CHECKED); /*Uncheck the previous radio button*/
    lv_obj_add_state(act_cb, LV_STATE_CHECKED); /*Check the current radio button*/
}
```

(continues on next page)

(continua dalla pagina precedente)

```

*active_id = lv_obj_get_index(act_cb);
LV_LOG_USER("Selected radio buttons: %d, %d", (int)active_index_1, (int)active_index_2);
}

static void radiobutton_create(lv_obj_t * parent, const char * txt)
{
    lv_obj_t * obj = lv_checkbox_create(parent);
    lv_checkbox_set_text(obj, txt);
    lv_obj_add_flag(obj, LV_OBJ_FLAG_EVENT_BUBBLE);
    lv_obj_add_style(obj, &style_radio, LV_PART_INDICATOR);
    lv_obj_add_style(obj, &style_radio_chk, LV_PART_INDICATOR | LV_STATE_CHECKED);
}

/**
 * Checkboxes as radio buttons
 */
void lv_example_checkbox_2(void)
{
    /* The idea is to enable `LV_OBJ_FLAG_EVENT_BUBBLE` on checkboxes and process the
     * `LV_EVENT_CLICKED` on the container.
     * A variable is passed as event user data where the index of the active
     * radiobutton is saved */

    lv_style_init(&style_radio);
    lv_style_set_radius(&style_radio, LV_RADIUS_CIRCLE);

    lv_style_init(&style_radio_chk);
    lv_style_set_bg_image_src(&style_radio_chk, NULL);

    uint32_t i;
    char buf[32];

    lv_obj_t * cont1 = lv_obj_create(lv_screen_active());
    lv_obj_set_flex_flow(cont1, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_size(cont1, lv_pct(40), lv_pct(80));
    lv_obj_add_event_cb(cont1, radio_event_handler, LV_EVENT_CLICKED, &active_index_1);

    for(i = 0; i < 5; i++) {
        lv_snprintf(buf, sizeof(buf), "A %d", (int)i + 1);
        radiobutton_create(cont1, buf);
    }

    /*Make the first checkbox checked*/
    lv_obj_add_state(lv_obj_get_child(cont1, 0), LV_STATE_CHECKED);

    lv_obj_t * cont2 = lv_obj_create(lv_screen_active());
    lv_obj_set_flex_flow(cont2, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_size(cont2, lv_pct(40), lv_pct(80));
    lv_obj_set_x(cont2, lv_pct(50));
    lv_obj_add_event_cb(cont2, radio_event_handler, LV_EVENT_CLICKED, &active_index_2);

    for(i = 0; i < 3; i++) {
        lv_snprintf(buf, sizeof(buf), "B %d", (int)i + 1);
        radiobutton_create(cont2, buf);
    }

    /*Make the first checkbox checked*/
    lv_obj_add_state(lv_obj_get_child(cont2, 0), LV_STATE_CHECKED);
}

#endif

```

## 6.12.7 API

*lv\_checkbox.h*

*lv\_checkbox\_private.h*

## 6.13 Drop-Down List (lv\_dropdown)

### 6.13.1 Panoramica

La Drop-Down List [elenco a discesa] consente all'utente di selezionare un valore da un elenco.

La Drop-Down List è chiusa per default e visualizza un singolo valore o un testo predefinito. Quando viene attivata (cliccando sulla Drop-Down List), viene creato un elenco da cui l'utente può selezionare un elemento. Quando l'utente seleziona un nuovo valore, l'elenco viene nuovamente eliminato.

La lista Drop-down viene aggiunta al gruppo di default (se impostato). La Drop-down list è un widget modificabile che consente la selezione degli elementi dell'elenco anche tramite codificatore o navigazione da tastiera.

### 6.13.2 Parti e Stili

Il Widget Drop-Down List è costituito dagli elementi: "button" e "list" (versioni semplificate dei Widget Button e List).

## Pulsante

- ***LV\_PART\_MAIN*** Background del button, usa le proprietà di stile *tipiche del background* e del testo per il suo testo.
- ***LV\_PART\_INDICATOR*** In genere un simbolo a forma di freccia che può essere un'immagine o un testo (ad esempio **LV\_SYMBOL**).

Il pulsante passa a ***LV\_STATE\_CHECKED*** quando viene aperto.

## Lista

- ***LV\_PART\_MAIN*** La lista stessa; utilizza le *tipiche proprietà di stile del background*. ***max\_height*** può essere utilizzato per limitare l'altezza della lista.
- ***LV\_PART\_SCROLLBAR*** Le proprietà di background, bordo, ombra e larghezza della barra di scorrimento (per la sua larghezza) e il padding destro per la spaziatura a destra.
- ***LV\_PART\_SELECTED*** Si riferisce all'opzione attualmente premuta, checked o premuta+checked. Utilizza anche le *tipiche proprietà di stile del background*.

L'elenco viene mostrato/nascosto all'apertura/chiusura. Per aggiungere stili, si usa ***lv\_dropdown\_get\_list***(dropdown) per ottenere l'oggetto elenco. Esempio:

```
lv_obj_t * list = lv_dropdown_get_list(dropdown) /* Get list */
lv_obj_add_style(list, &my_style, selector) /* Add styles to list */
```

In alternativa, il tema può essere esteso con nuovi stili.

### 6.13.3 Utilizzo

#### Elementi dell'elenco

Gli elementi dell'elenco vengono passati alla Drop-Down List come elenco "newline-separated" in una stringa come argomento **options** a ***lv\_dropdown\_set\_options***(dropdown, options). Ogni elemento dell'elenco deve essere separato da \n. Esempio: "First\nSecond\nThird". Questa stringa viene copiata dalla Drop-Down List, quindi il suo contenuto non deve rimanere disponibile oltre questa chiamata.

La funzione ***lv\_dropdown\_add\_option***(dropdown, "New option", pos) inserisce una nuova opzione all'indice **pos**.

Per risparmiare memoria, le opzioni possono essere impostate anche da una stringa statica (const) con ***lv\_dropdown\_set\_options\_static***(dropdown, options). In questo caso, il contenuto della stringa delle opzioni deve rimanere disponibile per tutta la durata della Drop-Down List e ***lv\_dropdown\_add\_option()*** non può essere utilizzato.

È possibile selezionare un'opzione a livello di codice con ***lv\_dropdown\_set\_selected***(dropdown, id), dove **id** è l'indice dell'opzione target.

#### Prelevare l'opzione selezionata

Per ottenere l'*indice* dell'opzione selezionata, si usa ***lv\_dropdown\_get\_selected***(dropdown).

***lv\_dropdown\_get\_selected\_str***(dropdown, buf, buf\_size) copia il *nome* dell'opzione selezionata in ``buf``.

#### Direzione

L'elenco può essere creato su qualsiasi lato. Il default ***LV\_DIR\_BOTTOM*** può essere modificato con ***lv\_dropdown\_set\_dir***(dropdown, ***LV\_DIR\_LEFT***).

Se l'elenco dovesse trovarsi verticalmente fuori dallo schermo, verrà allineato al bordo.

## Simbolo

Un simbolo (in genere una freccia) può essere aggiunto alla Drop-Down List con `lv_dropdown_set_symbol`(dropdown, LV\_SYMBOL\_...)

Se la direzione della Drop-Down List è `LV_DIR_LEFT` il simbolo verrà visualizzato a sinistra, altrimenti a destra.

## Mostrare il selezionato

La parte principale può mostrare l'elemento selezionato o un testo statico. Se statico è impostato con `lv_dropdown_set_text`(dropdown, "Some text"), verrà visualizzato indipendentemente dall'elemento selezionato. Se il testo è NULL, l'opzione selezionata viene visualizzata sul pulsante.

## Apertura/chiusura programmatica

Per aprire o chiudere programmaticamente la Drop-Down List si usa `lv_dropdown_open`(dropdown) o `lv_dropdown_close`(dropdown).

### 6.13.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando viene selezionata una nuova opzione o l'elenco viene aperto/chiuso.
- `LV_EVENT_CANCEL` Inviato quando l'elenco viene chiuso.
- `LV_EVENT_READY` Inviato quando l'elenco viene aperto.

#### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

### 6.13.5 Tasti

- `LV_KEY_RIGHT/DOWN` Seleziona l'elemento successivo dell'elenco.
- `LV_KEY_LEFT/UP` Seleziona l'elemento precedente dell'elenco.
- `LV_KEY_ENTER` Applica l'elemento dell'elenco selezionato (invia l'evento `LV_EVENT_VALUE_CHANGED` e chiude la Drop-Down List).

#### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

### 6.13.6 Esempio

#### Menù a discesa semplice

```
#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_dropdown_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Option: %s", buf);
    }
}

void lv_example_dropdown_1(void)
{
    /*Create a normal drop down list*/
    lv_obj_t * dd = lv_dropdown_create(lv_screen_active());
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_dropdown_set_options(dd, "Apple\n"
                        "Banana\n"
                        "Orange\n"
                        "Cherry\n"
                        "Grape\n"
                        "Raspberry\n"
                        "Melon\n"
                        "Orange\n"
                        "Lemon\n"
                        "Nuts");

lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 20);
lv_obj_add_event_cb(dd, event_handler, LV_EVENT_ALL, NULL);
}

#endif

```

## Menù a discesa nelle quattro direzioni

```

#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

/***
 * Create a drop down, up, left and right menus
 */
void lv_example_dropdown_2(void)
{
    static const char * opts = "Apple\n"
                              "Banana\n"
                              "Orange\n"
                              "Melon";

    lv_obj_t * dd;
    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 10);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_BOTTOM);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_UP);
    lv_obj_align(dd, LV_ALIGN_BOTTOM_MID, 0, -10);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_RIGHT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_RIGHT);
    lv_obj_align(dd, LV_ALIGN_LEFT_MID, 10, 0);

    dd = lv_dropdown_create(lv_screen_active());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_LEFT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_LEFT);
    lv_obj_align(dd, LV_ALIGN_RIGHT_MID, -10, 0);
}

#endif

```

## Menù

```

#include "../../lv_examples.h"
#if LV_USE_DROPDOWN && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * dropdown = lv_event_get_target_obj(e);
    char buf[64];
    lv_dropdown_get_selected_str(dropdown, buf, sizeof(buf));
    LV_LOG_USER("%s is selected", buf);
}

/***
 * Create a menu from a drop-down list and show some drop-down list features and styling
 */
void lv_example_dropdown_3(void)
{
    /*Create a drop down list*/
    lv_obj_t * dropdown = lv_dropdown_create(lv_screen_active());
    lv_obj_align(dropdown, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_dropdown_set_options(dropdown, "New project\n"
                           "New file\n"
                           "Save\n"
                           "Save as ..\n"
                           "Open project\n"
                           "Recent projects\n"
                           "Preferences\n"
                           "Exit");

    /*Set a fixed text to display on the button of the drop-down list*/
    lv_dropdown_set_text(dropdown, "Menu");

    /*Use a custom image as down icon and flip it when the list is opened*/
    LV_IMAGE_DECLARE(img_caret_down);
    lv_dropdown_set_symbol(dropdown, &img_caret_down);
    lv_obj_set_style_transform_rotation(dropdown, 1800, LV_PART_INDICATOR | LV_STATE_CHECKED);

    /*In a menu we don't need to show the last clicked item*/
    lv_dropdown_set_selected_highlight(dropdown, false);

    lv_obj_add_event_cb(dropdown, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
}

#endif

```

## 6.13.7 API

*lv\_api\_map\_v8.h*  
*lv\_observer.h*  
*lv\_dropdown.h*  
*lv\_dropdown\_private.h*  
*lv\_obj\_property\_names.h*

## 6.14 Image (lv\_image)

### 6.14.1 Panoramica

Le immagini sono Widget che visualizzano immagini da Flash (come array) o da file. Le immagini possono anche visualizzare simboli (LV\_SYMBOL\_...).

Utilizzando *l'interfaccia di decodifica delle immagini*, è possibile supportare anche formati di immagine personalizzati.

### 6.14.2 Parti e Stili

- **LV\_PART\_MAIN** Un rettangolo di background che utilizza le *tipiche proprietà di stile del background*, e l'immagine stessa utilizza le proprietà di stile dell'immagine.

### 6.14.3 Utilizzo

#### Sorgente dell'immagine

Per garantire la massima flessibilità, la sorgente dell'immagine può essere:

- una variabile nel codice (un array C contenente i pixel).
- un file memorizzato esternamente (ad esempio su una scheda SD).
- un *Symbol* come testo.

Per impostare la sorgente di un'immagine, si usa *lv\_image\_set\_src(img, src)*.

Per generare un array di pixel da un'immagine PNG, JPG o BMP, utilizzare il [Tool di conversione di immagini online](#) e impostare l'immagine convertita come sorgente dell'immagine con il suo puntatore tramite *lv\_image\_set\_src(img1, &converted\_img\_var)*. Per rendere accessibile la variabile immagine convertita dal file C, dichiararla con *LV\_IMAGE\_DECLARE(converted\_img\_var)*.

Per utilizzare file esterni, è necessario convertire i file immagine utilizzando lo strumento di conversione online, ma selezionando il formato di output binario. È inoltre necessario utilizzare il modulo file system di LVGL e registrare un driver con alcune funzioni per le operazioni di base sui file. Per ulteriori informazioni: [File system](#). Quindi impostare l'immagine tradotta come sorgente dell'immagine con *lv\_image\_set\_src(img, "S:folder1/my\_img.bin")*.

È anche possibile impostare un simbolo come sorgente dell'immagine, in modo simile a [Labels](#). In questo caso, l'immagine verrà visualizzata come testo in base al *font* specificato nello stile. Permette l'uso di "caratteri" monocromatici leggeri al posto di immagini reali. È possibile impostare un simbolo come sorgente dell'immagine con *lv\_image\_set\_src(img1, LV\_SYMBOL\_OK)*.

#### Etichetta come immagine

Immagini ed etichette vengono talvolta utilizzate per comunicare la stessa cosa, ad esempio per descrivere la funzione di un pulsante. In questo contesto, immagini ed etichette sono in qualche modo intercambiabili: le immagini possono visualizzare del testo utilizzando la macro **LV\_SYMBOL\_DUMMY** (che equivale a una stringa C di 3 byte contenente un codice speciale) come prefisso del testo. Ad esempio, *lv\_image\_set\_src(img, LV\_SYMBOL\_DUMMY "Some text")*.

## Trasparenza

Le immagini interne (array di pixel) ed esterne supportano 2 metodi di gestione della trasparenza:

- **Byte alpha:** Un canale alfa viene aggiunto a ogni pixel che ne contiene l'opacità, in genere un byte. È la 'A' nei vari formati colore che contengono un canale alfa, come ARGB8888, ARGB8565, ARGB1555, ecc.
- **Colore trasparente indicizzato:** un indice specifico in una tavolozza di colori serve a segnalare la trasparenza per ogni pixel che lo utilizza.

## Palette e indice Alfa

Oltre al formato colore *True color* (RGB), sono supportati i seguenti formati:

- **Indicizzato:** L'immagine ha una tavolozza colori e ogni pixel è un indice di tale tavolozza.
- **Indicizzato alfa:** I valori memorizzati nelle posizioni dei pixel sono valori alfa (opacità).

Queste opzioni possono essere selezionate nel convertitore di immagini. Per saperne di più sui formati colore, consultare la sezione [Color formats](#).

## Recolor

Un colore può essere miscelato con ogni pixel di un'immagine con una determinata intensità. Questo può essere utile per mostrare diversi stati (selezionato, inattivo, premuto, ecc.) di un'immagine senza dover memorizzare più versioni della stessa immagine. Questa funzionalità può essere abilitata nello stile impostando `img_recolor_opa` tra `LV_OPA_TRANSPI` (nessuna ricolorazione, valore: 0) e `LV_OPA_COVER` (ricolorazione completa, valore: 255). Il valore di default è `LV_OPA_TRANSPI`, che disabilita questa funzionalità.

Il colore da miscelare è impostato da `img_recolor`.

## Offset

Con `lv_image_set_offset_x(img, x_ofs)` e `lv_image_set_offset_y(img, y_ofs)`, è possibile aggiungere un offset all'immagine visualizzata. Utile se le dimensioni del Widget sono inferiori a quelle dell'immagine sorgente. Utilizzando il parametro offset è possibile creare una [Texture atlas](#) o un effetto "immagine in movimento" [Animando](#) l'offset x o y.

## Trasformazioni

È possibile ingrandire o ridurre le immagini utilizzando `lv_image_set_scale(img, factor)`. Impostare `factor` a 256 o a `LV_SCALE_NONE` per disabilitare lo zoom. Un valore maggiore ingrandisce le immagini (ad esempio 512 raddoppiandone le dimensioni), un valore minore le rimpicciolisce (ad esempio 128 dimezzandone le dimensioni). Il ridimensionamento frazionario funziona utilizzando un valore proporzionalmente maggiore o minore, ad esempio 281 per un ingrandimento del 10%.

`lv_image_set_scale_x(img, factor)` e `lv_image_set_scale_y(img, factor)` possono essere utilizzati anche per impostare il ridimensionamento orizzontale e verticale in modo indipendente. Possono essere valori diversi.

Per ruotare l'immagine, si usa `lv_image_set_rotation(img, angle_x10)`. L'argomento `angle_x10` è un `int32_t` contenente l'angolo (in gradi) moltiplicato per 10. Questo fornisce una risoluzione di 0,1 gradi. Esempio: 458 significa 45,8°.

Per default il punto di rotazione è il centro dell'immagine. Questo può essere modificato con `lv_image_set_pivot(img, pivot_x, pivot_y)` dove le coordinate (0,0) rappresentano l'angolo in alto a sinistra.

La qualità della trasformazione può essere regolata con `lv_image_set_antialias(img, true)`. L'attivazione dell'anti-aliasing rende le trasformazioni di qualità superiore, ma più lente.

Le trasformazioni richiedono che l'intera immagine sia disponibile. Pertanto, le immagini indicizzate (`LV_COLOR_FORMAT_I1/2/4/8...`) e le immagini solo alfa non possono essere trasformate. In altre parole, le trasformazioni funzionano solo su immagini normali (A)RGB o A8 memorizzate come array C, o su immagini fornite da un [Image decoder](#) personalizzato che restituisce l'intera immagine.

Si noti che le coordinate reali dei Widget immagine non cambiano con una trasformazione. Ciò significa che `lv_obj_get_width / height / x / y()` restituirà le coordinate originali, non ingrandite.

**IMPORTANTE:** La trasformazione dell'immagine è indipendente dalle proprietà di trasformazione *derivate dagli stili*. Le principali differenze sono che le trasformazioni pure del Widget Image:

- non trasformano gli elementi figlio del Widget immagine e
- l'immagine viene trasformata direttamente senza creare un livello intermedio (buffer) per catturare l'immagine del Widget.

## Allineamento interno

Per default, la larghezza e l'altezza del Widget "image" sono `LV_SIZE_CONTENT`, il che significa che il widget verrà ridimensionato automaticamente in base alle dimensioni dell'immagine sorgente.

Se la larghezza o l'altezza del Widget sono impostate su un valore diverso, il valore della proprietà `inner_align` (impostato tramite `lv_image_set_inner_align(widget, align)`) determina l'allineamento dell'immagine sorgente all'interno del Widget.

`align` può assumere uno qualsiasi di questi valori:

- `LV_IMAGE_ALIGN_DEFAULT`: Significa in alto a sinistra
- `LV_IMAGE_ALIGN_TOP_LEFT`
- `LV_IMAGE_ALIGN_TOP_MID`
- `LV_IMAGE_ALIGN_TOP_RIGHT`
- `LV_IMAGE_ALIGN_BOTTOM_LEFT`
- `LV_IMAGE_ALIGN_BOTTOM_MID`
- `LV_IMAGE_ALIGN_BOTTOM_RIGHT`
- `LV_IMAGE_ALIGN_LEFT_MID`
- `LV_IMAGE_ALIGN_RIGHT_MID`
- `LV_IMAGE_ALIGN_CENTER`
- `LV_IMAGE_ALIGN_STRETCH`
- `LV_IMAGE_ALIGN_TILE`
- `LV_IMAGE_ALIGN_CONTAIN`
- `LV_IMAGE_ALIGN_COVER`

qualsiasi valore di `offset` viene applicato dopo l'allineamento della sorgente dell'immagine. Ad esempio, impostando un offset di `y=-10` con `align==LV_IMAGE_ALIGN_CENTER`, la sorgente dell'immagine verrà spostata di 10 pixel verso l'alto rispetto al centro del Widget.

Per ridimensionare o affiancare automaticamente l'immagine, passare uno di questi valori di `align`:

- `LV_IMAGE_ALIGN_STRETCH` Imposta la scala X e Y per riempire l'area del Widget.
- `LV_IMAGE_ALIGN_TILE` Affianca l'immagine per riempire l'area del Widget. L'offset viene applicato per spostare l'affiancamento.
- `LV_IMAGE_ALIGN_CONTAIN` L'immagine mantiene le sue proporzioni, ma viene ridimensionata alla dimensione massima che rientra nell'area del Widget.
- `LV_IMAGE_ALIGN_COVER` L'immagine mantiene le sue proporzioni e riempie l'area del Widget.

## 6.14.4 Eventi

I Widget immagine non inviano eventi speciali. Per default, i Widget Image vengono creati senza il flag LV\_OBJ\_FLAG\_CLICKABLE, ma è possibile aggiungerlo per far sì che un Widget immagine rilevi ed emetta eventi LV\_EVENT\_CLICKED, se lo si desidera.

### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

## 6.14.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Image.

### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

## 6.14.6 Esempio

### Immagine da variabile e simbolo

```
#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES
void lv_example_image_1(void)
{
    LV_IMAGE_DECLARE(img_cogwheel_argb);
    lv_obj_t * img1 = lv_image_create(lv_screen_active());
    lv_image_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_CENTER, 0, 0);

    lv_obj_t * img2 = lv_image_create(lv_screen_active());
    lv_image_set_src(img2, LV_SYMBOL_OK "Accept");
    lv_obj_align_to(img2, img1, LV_ALIGN_OUT_BOTTOM_MID, 0, 20);
}
#endif
```

### Ricolorazione dell'immagine

```
#include "../../lv_examples.h"
#if LV_USE_IMAGE && LV_USE_SLIDER && LV_BUILD_EXAMPLES
static lv_obj_t * create_slider(lv_color_t color);
static void slider_event_cb(lv_event_t * e);

static lv_obj_t * red_slider, * green_slider, * blue_slider, * intense_slider;
static lv_obj_t * img1;

/***
 * Demonstrate runtime image re-coloring
 */
void lv_example_image_2(void)
{
    /*Create 4 sliders to adjust RGB color and re-color intensity*/
    red_slider = create_slider(lv_palette_main(LV_PALETTE_RED));
    green_slider = create_slider(lv_palette_main(LV_PALETTE_GREEN));
    blue_slider = create_slider(lv_palette_main(LV_PALETTE_BLUE));
    intense_slider = create_slider(lv_palette_main(LV_PALETTE_GREY));

    lv_slider_set_value(red_slider, LV_OPA_20, LV_ANIM_OFF);
    lv_slider_set_value(green_slider, LV_OPA_90, LV_ANIM_OFF);
    lv_slider_set_value(blue_slider, LV_OPA_60, LV_ANIM_OFF);
    lv_slider_set_value(intense_slider, LV_OPA_50, LV_ANIM_OFF);

    lv_obj_align(red_slider, LV_ALIGN_LEFT_MID, 25, 0);
    lv_obj_align_to(green_slider, red_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(blue_slider, green_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);
    lv_obj_align_to(intense_slider, blue_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);

    /*Now create the actual image*/
    LV_IMAGE_DECLARE(img_cogwheel_argb);
    img1 = lv_image_create(lv_screen_active());
    lv_image_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_RIGHT_MID, -20, 0);
    lv_obj_send_event(intense_slider, LV_EVENT_VALUE_CHANGED, NULL);
}

static void slider_event_cb(lv_event_t * e)
{
```

(continues on next page)

(continua dalla pagina precedente)

```

LV_UNUSED(e);

/*Recolor the image based on the sliders' values*/
lv_color_t color = lv_color_make((uint8_t)lv_slider_get_value(red_slider),
                                  (uint8_t)lv_slider_get_value(green_slider),
                                  (uint8_t)lv_slider_get_value(blue_slider));
lv_opa_t intense = (lv_opa_t)lv_slider_get_value(intense_slider);
lv_obj_set_style_image_recolor_opa(img1, intense, 0);
lv_obj_set_style_image_recolor(img1, color, 0);
}

static lv_obj_t * create_slider(lv_color_t color)
{
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_slider_set_range(slider, 0, 255);
    lv_obj_set_size(slider, 10, 200);
    lv_obj_set_style_bg_color(slider, color, LV_PART_KNOB);
    lv_obj_set_style_bg_color(slider, lv_color_darken(color, LV_OPA_40), LV_PART_INDICATOR);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    return slider;
}
#endif

```

## Rotazione e zoom

```

#include "../../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES

static void set_angle(void * img, int32_t v)
{
    lv_image_set_rotation((lv_obj_t *)img, v);
}

static void set_scale(void * img, int32_t v)
{
    lv_image_set_scale((lv_obj_t *)img, v);
}

/**
 * Show transformations (zoom and rotation) using a pivot point.
 */
void lv_example_image_3(void)
{
    LV_IMAGE_DECLARE(img_cogwheel_argb);

    /*Now create the actual image*/
    lv_obj_t * img = lv_image_create(lv_screen_active());
    lv_image_set_src(img, &img_cogwheel_argb);
    lv_obj_align(img, LV_ALIGN_CENTER, 50, 50);
    lv_image_set_pivot(img, 0, 0); /*Rotate around the top left corner*/

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, img);
    lv_anim_set_exec_cb(&a, set_angle);
    lv_anim_set_values(&a, 0, 3600);
    lv_anim_set_duration(&a, 5000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, set_scale);
    lv_anim_set_values(&a, 128, 256);
    lv_anim_set_reverse_duration(&a, 3000);
    lv_anim_start(&a);
}

#endif

```

## Offset e stile dell'immagine

```

#include "../../../lv_examples.h"
#if LV_USE_IMAGE && LV_BUILD_EXAMPLES

static void ofs_y_anim(void * img, int32_t v)
{
    lv_image_set_offset_y((lv_obj_t *)img, v);
}

/**
 * Image styling and offset
 */
void lv_example_image_4(void)
{
    LV_IMAGE_DECLARE(img_skew_strip);

    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_image_recolor_opa(&style, LV_OPA_COVER);
    lv_style_set_image_recolor(&style, lv_color_black());

    lv_obj_t * img = lv_image_create(lv_screen_active());
    lv_obj_add_style(img, &style, 0);
    lv_image_set_src(img, &img_skew_strip);
    lv_obj_set_size(img, 150, 100);
    lv_obj_center(img);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, img);
    lv_anim_set_exec_cb(&a, ofs_y_anim);
    lv_anim_set_values(&a, 0, 100);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_anim_set_duration(&a, 3000);
lv_anim_set_reverse_duration(&a, 500);
lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
lv_anim_start(&a);
}

#endif

```

## 6.14.7 API

*lv\_draw\_buf.h*  
*lv\_draw\_image.h*  
*lv\_draw\_image\_private.h*  
*lv\_image\_decoder.h*  
*lv\_image\_decoder\_private.h*  
*lv\_image\_dsc.h*  
*lv\_api\_map\_v8.h*  
*lv\_api\_map\_v9\_0.h*  
*lv\_api\_map\_v9\_1.h*  
*lv\_image\_cache.h*  
*lv\_image\_header\_cache.h*  
*lv\_image.h*  
*lv\_image\_private.h*  
*lv\_imagebutton.h*  
*lv\_imagebutton\_private.h*  
*lv\_obj\_property\_names.h*

## 6.15 Image Button (lv\_imagebutton)

### 6.15.1 Panoramica

"Image Button" [pulsante immagine] è molto simile al semplice Widget 'Button'. L'unica differenza è che visualizza immagini definite dall'utente per ogni stato invece di disegnare un rettangolo. L'elenco degli stati è illustrato di seguito.

È possibile impostare un'immagine a sinistra, a destra e al centro; l'immagine al centro verrà ripetuta per adattarsi alla larghezza del widget.

### 6.15.2 Parti e Stili

- *LV\_PART\_MAIN* Si riferisce all'immagine/i. Se vengono utilizzate le proprietà di stile del background, verrà disegnato un rettangolo dietro l'Image Button.

### 6.15.3 Utilizzo

#### Sorgenti delle immagini

Per impostare l'immagine in uno stato, utilizzare *lv\_imagebutton\_set\_src(imagebutton, LV\_IMAGEBUTTON\_STATE\_..., src\_left, src\_mid, src\_right)*.

Le sorgenti delle immagini funzionano come descritto nel [Widget Image](#) con la differenza che i "Simboli" non sono supportati dall'Image Button. Qualsiasi sorgente può essere NULL. In genere, l'immagine centrale dovrebbe essere una delle sorgenti immagini impostate.

Se viene specificato solo `src_mid`, la larghezza del widget verrà impostata automaticamente sulla larghezza dell'immagine. Tuttavia, se vengono impostate tutte e tre le sorgenti, la larghezza deve essere impostata dall'utente (ad esempio, utilizzando `lv_obj_set_width`) e l'immagine centrale verrà suddivisa in riquadri per riempire la dimensione specificata.

I possibili stati sono:

- `LV_IMAGEBUTTON_STATE_RELEASED`
- `LV_IMAGEBUTTON_STATE_PRESSED`
- `LV_IMAGEBUTTON_STATE_DISABLED`
- `LV_IMAGEBUTTON_STATE_CHECKED_RELEASED`
- `LV_IMAGEBUTTON_STATE_CHECKED_PRESSED`
- `LV_IMAGEBUTTON_STATE_CHECKED_DISABLED`

Le sorgenti immagine impostate per lo stato `LV_IMAGEBUTTON_STATE_RELEASED` vengono utilizzate per qualsiasi stato per il quale non siano state impostate sorgenti immagine. Se sono state impostate sorgenti immagine per altri stati, ad esempio `LV_IMAGEBUTTON_STATE_PRESSED`, queste verranno utilizzate al loro posto quando l'Image Button si trova in quello stato.

#### Impostazione dello Stato Tramite Programmazione

Invece delle normali funzioni `lv_obj_add_state()` and `lv_obj_remove_state()`, si usa `lv_imagebutton_set_state(imagebutton, LV_IMAGEBUTTON_STATE_...)` per impostare lo stato degli Image Button.

#### 6.15.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Invia quando lo stato CHECKED dell'Image Button viene attivato/disattivato. Richiede l'impostazione del flag `LV_OBJ_FLAG_CHECKABLE` dell'Image Button.

##### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

#### 6.15.5 Tasti

- `LV_KEY_RIGHT/UP` Passa allo stato CHECKED se `LV_OBJ_FLAG_CHECKABLE` è abilitato.
- `LV_KEY_LEFT/DOWN` Passa allo stato non CHECKED se `LV_OBJ_FLAG_CHECKABLE` è abilitato.
- `LV_KEY_ENTER` Clicca sull'Image Button

##### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

#### 6.15.6 Esempio

##### Pulsante immagine semplice

```
#include "../../lv_examples.h"
#if LV_USE_IMAGEBUTTON && LV_BUILD_EXAMPLES
void lv_example_imagebutton_1(void)
{
    LV_IMAGE_DECLARE(imagebutton_left);
    LV_IMAGE_DECLARE(imagebutton_right);
    LV_IMAGE_DECLARE(imagebutton_mid);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

/*Create a transition animation on width transformation and recolor.*/
static lv_style_prop_t tr_prop[] = {LV_STYLE_TRANSFORM_WIDTH, LV_STYLE_IMAGE_RECOLOR_OPA, 0};
static lv_style_transition_dsc_t tr;
lv_style_transition_dsc_init(&tr, tr_prop, lv_anim_path_linear, 200, 0, NULL);

static lv_style_t style_def;
lv_style_init(&style_def);
lv_style_set_text_color(&style_def, lv_color_white());
lv_style_set_transition(&style_def, &tr);

/*Darken the button when pressed and make it wider*/
static lv_style_t style_pr;
lv_style_init(&style_pr);
lv_style_set_image_recolor_opa(&style_pr, LV_OPA_30);
lv_style_set_image_recolor(&style_pr, lv_color_black());
lv_style_set_transform_width(&style_pr, 20);

/*Create an image button*/
lv_obj_t * imagebutton1 = lv_imagebutton_create(lv_screen_active());
lv_imagebutton_set_src(imagebutton1, LV_IMAGEBUTTON_STATE_RELEASED, &imagebutton_left, &imagebutton_mid,
                      &imagebutton_right);
lv_obj_add_style(imagebutton1, &style_def, 0);
lv_obj_add_style(imagebutton1, &style_pr, LV_STATE_PRESSED);

lv_obj_set_width(imagebutton1, 100);
lv_obj_align(imagebutton1, LV_ALIGN_CENTER, 0, 0);

/*Create a label on the image button*/
lv_obj_t * label = lv_label_create(imagebutton1);
lv_label_set_text(label, "Button");
lv_obj_align(label, LV_ALIGN_CENTER, 0, -4);
}

#endif

```

## 6.15.7 API

*lv\_imagebutton.h*

*lv\_imagebutton\_private.h*

# 6.16 Keyboard (lv\_keyboard)

## 6.16.1 Panoramica

Il Widget "Keyboard" [tastiera] è una speciale *Button Matrix* (*lv\_buttonmatrix*) con mappe dei tasti predefinite e altre funzionalità per fornire una tastiera virtuale su schermo per scrivere testo in una *Text Area* (*lv\_textarea*).

## 6.16.2 Parti e Stili

Simile alla Button Matrix, il Widget Keyboard è composto da 2 parti:

- *LV\_PART\_MAIN* La parte principale. Utilizza le *tipiche proprietà di stile del background*
- *LV\_PART\_ITEMS* I pulsanti. Utilizza anche le *tipiche proprietà di stile del background* e le proprietà *text*.

## 6.16.3 Utilizzo

### Modes

Le tastiere hanno le seguenti modalità:

- *LV\_KEYBOARD\_MODE\_TEXT\_LOWER* Visualizza lettere minuscole
- *LV\_KEYBOARD\_MODE\_TEXT\_UPPER* Visualizza lettere maiuscole
- *LV\_KEYBOARD\_MODE\_SPECIAL* Visualizza caratteri speciali
- *LV\_KEYBOARD\_MODE\_NUMBER* Visualizza numeri, segno +/- e punto decimale
- Da *LV\_KEYBOARD\_MODE\_USER\_1* a *LV\_KEYBOARD\_MODE\_USER\_4* Modalità definite dall'utente.

I layout delle modalità TEXT contengono "tasti" per cambiare modalità.

Per impostare la modalità a livello di codice, utilizzare *lv\_keyboard\_set\_mode(kb, mode)*. La modalità di default è *LV\_KEYBOARD\_MODE\_TEXT\_UPPER*.

## Assegnazione dell'Area di Testo

È possibile assegnare un'*Area di testo* alla tastiera per posizionarvi automaticamente i caratteri cliccati. Per assegnare la Text Area, si usa `lv_keyboard_set_textarea(kb, text_area)`.

## Pop-over dei Tasti

Per abilitare i pop-over dei tasti alla pressione, come sulle comuni tastiere Android e iOS, si usa `lv_keyboard_set_popovers(kb, true)`. Le mappe di controllo di default sono preconfigurate per mostrare i pop-over solo sui tasti che producono un simbolo (ovvero non sulla barra spaziatrice). Se si utilizza una mappa di controllo personalizzata (vedere di seguito), impostare il flag `LV_BUTTONMATRIX_CTRL_POPOVER` per ogni tasto per cui si desidera visualizzare un pop-over.

Si noti che i pop-over per i tasti nella riga superiore verranno visualizzati al di fuori dei limiti del widget. Per risolvere questo problema, riservare spazio libero extra sopra la Keyboard o assicurarsi che questa venga aggiunta *dopo* eventuali Widget adiacenti al suo limite superiore (posizionandola "sopra" tali Widget) in modo che i pop-over vengano visualizzati sopra di essi.

Al momento, i pop-over sono solo un effetto visivo e non consentono ancora la selezione di caratteri aggiuntivi, come quelli accentati.

## Nuova Mappa dei Pulsanti

È possibile specificare una nuova mappa (layout) per la tastiera con `lv_keyboard_set_map(kb, LV_KEYBOARD_MODE_..., kb_map, kb_ctrl)`. Consultare la sezione *Mappa dei pulsanti* del Button Matrix per ulteriori informazioni sulla creazione di nuove mappe.

Tenere presente che l'utilizzo delle seguenti parole chiave nella mappa avrà lo stesso effetto della mappa originale:

- `LV_SYMBOL_OK` Invia `LV_EVENT_READY` alla Text Area assegnata.
- `LV_SYMBOL_CLOSE` o `LV_SYMBOL_KEYBOARD` Inviano `LV_EVENT_CANCEL` all'area di testo assegnata.
- `LV_SYMBOL_BACKSPACE` Cancella il carattere a sinistra.
- `LV_SYMBOL_LEFT` Sposta il cursore a sinistra.
- `LV_SYMBOL_RIGHT` Sposta il cursore a destra.
- `LV_SYMBOL_NEW_LINE` Nuova riga.
- "ABC" Carica la mappa delle maiuscole.
- "abc" Carica la mappa delle minuscole.
- "1#" Carica la mappa dei numeri.

## Personalizzazione dei tasti

Ogni tasto può essere personalizzato utilizzando eventi di disegno come `LV_EVENT_DRAW_TASK_ADDED`.

Controllare gli esempi per vedere come può essere utilizzato nella pratica.

### 6.16.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando il pulsante viene premuto/rilasciato o ripetuto dopo una pressione prolungata. I dati dell'evento contengono l'ID del pulsante premuto/rilasciato.
- `LV_EVENT_READY`: È stato cliccato il pulsante *Ok*.
- `LV_EVENT_CANCEL`: È stato cliccato il pulsante *Close*.

La tastiera dispone di un **gestore eventi predefinito** chiamato `lv_keyboard_def_event_cb()`, che gestisce la pressione del pulsante, il cambio di mappa, l'invio di eventi all'area di testo assegnata, ecc. È possibile rimuoverlo e sostituirlo con un gestore eventi personalizzato, se lo si desidera, oppure aggiungerne uno personalizzato.

**i Nota**

In LVGL v8.0 e versioni successive, l'aggiunta di un gestore eventi alla Keyboard non rimuove il gestore eventi di default. Questo comportamento è diverso dalla versione 7, dove l'aggiunta di un gestore di eventi sostituirebbe quello precedente.

**i Ulteriori Informazioni**

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

**6.16.5 Tasti**

- **LV\_KEY\_RIGHT/UP/LEFT/RIGHT** Per navigare tra i pulsanti, selezionando quello desiderato.
- **LV\_KEY\_ENTER** Per premere/rilasciare il pulsante selezionato.

**i Ulteriori Informazioni**

Scoprire di più sui *Tasti*.

**6.16.6 Esempio****Tastiera con area di testo**

```
#include "../../lv_examples.h"
#if LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target_obj(e);
    lv_obj_t * kb = (lv_obj_t *)lv_event_get_user_data(e);
    if(code == LV_EVENT_FOCUSED) {
        lv_keyboard_set_textarea(kb, ta);
        lv_obj_remove_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }

    if(code == LV_EVENT_DEFOCUSSED) {
        lv_keyboard_set_textarea(kb, NULL);
        lv_obj_add_flag(kb, LV_OBJ_FLAG_HIDDEN);
    }
}

void lv_example_keyboard_1(void)
{
    /*Create a keyboard to use it with an of the text areas*/
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());

    /*Create a text area. The keyboard will write here*/
    lv_obj_t * ta1;
    ta1 = lv_textarea_create(lv_screen_active());
    lv_obj_align(ta1, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_obj_add_event_cb(ta1, ta_event_cb, LV_EVENT_ALL, kb);
    lv_textarea_set_placeholder_text(ta1, "Hello");
    lv_obj_set_size(ta1, 140, 80);

    lv_obj_t * ta2;
    ta2 = lv_textarea_create(lv_screen_active());
    lv_obj_align(ta2, LV_ALIGN_TOP_RIGHT, -10, 10);
    lv_obj_add_event_cb(ta2, ta_event_cb, LV_EVENT_ALL, kb);
    lv_obj_set_size(ta2, 140, 80);

    lv_keyboard_set_textarea(kb, ta1);

    /*The keyboard will show Arabic characters if they are enabled */
#endif
}
#endif
```

## Tastiera con mappa personalizzata

```
#include "../../lv_examples.h"
#if LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

void lv_example_keyboard_2(void)
{
    /*Create an AZERTY keyboard map*/
    static const char * kb_map[] = {"A", "Z", "E", "R", "T", "Y", "U", "I", "O", "P", LV_SYMBOL_BACKSPACE, "\n",
                                    "Q", "S", "D", "F", "G", "J", "K", "L", "M", LV_SYMBOL_NEW_LINE, "\n",
                                    "W", "X", "C", "V", "B", "N", "H", "V", "I", "?", "\n",
                                    LV_SYMBOL_CLOSE, " ", " ", " ", LV_SYMBOL_OK, NULL
                                };

    /*Set the relative width of the buttons and other controls*/
    static const lv_buttonmatrix_ctrl_t kb_ctrl[] = {LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4,
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4,
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_6,
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4,
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4,
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4, LV_BUTTONMATRIX_CTRL_WIDTH_4,
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_2, (lv_buttonmatrix_ctrl_t)(LV_BUTTONMATRIX_CTRL_HIDDEN | LV_BUTTONMATRIX_CTRL_WIDTH_2),
    ↪LV_BUTTONMATRIX_CTRL_WIDTH_2
};

    /*Create a keyboard and add the new map as USER_1 mode*/
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());

    lv_keyboard_set_map(kb, LV_KEYBOARD_MODE_USER_1, kb_map, kb_ctrl);
    lv_keyboard_set_mode(kb, LV_KEYBOARD_MODE_USER_1);

    /*Create a text area. The keyboard will write here*/
    lv_obj_t * ta;
    ta = lv_textarea_create(lv_screen_active());
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, 10);
    lv_obj_set_size(ta, lv_pct(90), 80);
    lv_obj_add_state(ta, LV_STATE_FOCUSED);

    lv_keyboard_set_textarea(kb, ta);
}
#endif
```

## Tastiera con disegno

```
#include "../../lv_examples.h"
#if LV_USE_BUTTONMATRIX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_dsc(draw_task);
    bool pressed = false;
    if(lv_keyboard_get_selected_button(obj) == base_dsc->id1 && lv_obj_has_state(obj, LV_STATE_PRESSED)) {
        pressed = true;
    }

    /*When the keyboard draws the buttons...*/
    if(base_dsc->part == LV_PART_ITEMS) {
        /*Get a color based on the button's index*/
        lv_palette_t palette = (lv_palette_t)(base_dsc->id1 % LV_PALETTE_LAST);
        lv_draw_dsc_base_t * fill_dsc = lv_draw_task_get_fill_dsc(draw_task);
        if(fill_dsc) {
            fill_dsc->color = pressed ? lv_palette_darken(palette, 3) : lv_palette_main(palette);
        }
        lv_draw_label_dsc_t * label_dsc = lv_draw_task_get_label_dsc(draw_task);
        if(label_dsc) {
            /*For the OK symbol, draw a star instead*/
            if(lv_streq(label_dsc->text, LV_SYMBOL_OK) == 0) {
                label_dsc->sopa = 0; /*Hide the label*/
                LV_IMAGE_DECLARE(img_star);
                lv_image_header_t header;
                lv_result_t res = lv_image_decoder_get_info(&img_star, &header);
                if(res != LV_RESULT_OK) return;

                lv_area_t a;
                lv_area_set(&a, 0, 0, header.w - 1, header.h - 1);
                lv_area_t draw_task_area;
                lv_draw_task_get_area(draw_task, &draw_task_area);
                lv_area_align(&draw_task_area, &a, LV_ALIGN_CENTER, 0, 0);

                lv_draw_image_dsc_t img_draw_dsc;
                lv_draw_image_dsc_init(&img_draw_dsc);
                img_draw_dsc.src = &img_star;
                lv_draw_image(base_dsc->layer, &img_draw_dsc, &a);
            } else {
                /*For the other labels just pick an lighter color*/
                label_dsc->color = lv_palette_lighten(palette, 4);
            }
        }
    }
}

/**
 * Add custom drawer to the keyboard to customize the buttons one by one
 */
void lv_example_keyboard_3(void)
{
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());
    lv_obj_add_event_cb(kb, event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(kb, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    lv_obj_center(kb);
```

(continues on next page)

(continua dalla pagina precedente)

```
}
```

```
#endif
```

## 6.16.7 API

*lv\_api\_map\_v8.h*  
*lv\_keyboard.h*  
*lv\_keyboard\_private.h*  
*lv\_obj\_property\_names.h*

## 6.17 Label (lv\_label)

### 6.17.1 Panoramica

Una "Label" [etichetta] è il Widget utilizzato per visualizzare il testo.

### 6.17.2 Parti e Stili

- *LV\_PART\_MAIN* Utilizza le proprietà *typical background* e *text*. I valori di padding possono essere utilizzati per aggiungere spazio tra il testo e i bordi del background dell'etichetta.
- *LV\_PART\_SCROLLBAR* La barra di scorrimento che viene visualizzata quando il testo è più grande delle dimensioni del Widget.
- *LV\_PART\_SELECTED* Indica lo stile del *testo selezionato*. È possibile utilizzare solo le proprietà di stile *text\_color* e *bg\_color*.

### 6.17.3 Utilizzo

#### Impostazione del testo

È possibile impostare il testo su un'etichetta in fase di esecuzione con *lv\_label\_set\_text(label, "New text")*. Questo allocherà un buffer dinamicamente e la stringa fornita verrà copiata in quel buffer. Pertanto, non è necessario mantenere il testo passato a *lv\_label\_set\_text()* nello scope dopo il ritorno della funzione.

Con *lv\_label\_set\_text\_fmt(label, fmt, ...)* è possibile utilizzare la formattazione printf per impostare il testo. Esempio: *lv\_label\_set\_text\_fmt(label, "Value: %d", 15)*.

Le etichette possono visualizzare anche il testo da un buffer di caratteri statico. Per farlo, si usa *lv\_label\_set\_text\_static(label, "Text")*. In questo caso, il testo non viene posto nella memoria dinamica e viene utilizzato direttamente il buffer specificato. Ciò significa che il contenuto del buffer dei caratteri deve rimanere valido per tutta la durata dell'etichetta o finché non viene impostato un altro buffer tramite una delle funzioni sopra indicate.

Le stringhe **const** sono sicure da usare con *lv\_label\_set\_text\_static()* poiché sono memorizzate nella memoria ROM, che è sempre accessibile.

#### ⚠️ Avvertimento

Non utilizzare stringhe **const** con *lv\_label\_set\_text\_static()* quando l'etichetta viene utilizzata in modalità *LV\_LABEL\_LONG\_DOT* poiché l'etichetta tenterà di modificare la stringa sul posto. Ciò causerà un'eccezione MCU tentando di modificare la memoria di programma (ROM).

### Attenzione

Se l'etichetta viene aggiornata rapidamente con nuove stringhe (ad esempio > 30X/secondo, come RPM in una dashboard o un valore ADC) e la lunghezza di tali stringhe cambia frequentemente, è consigliabile:

- allocare un buffer di stringhe statico sufficientemente grande da contenere la stringa più grande possibile,
- aggiornare tale buffer con le nuove stringhe solo quando queste apporteranno una differenza visibile per l'utente finale e
- aggiornare l'etichetta con `lv_label_set_text_static(label, buffer)` utilizzando tale buffer.

Motivo: se si utilizza `lv_label_set_text(label, new_text)`, verrà forzata una realloc() di memoria ogni volta che la lunghezza della stringa cambia. Questo sovraccarico della MCU può essere evitato eseguendo quanto sopra.

### Newline

I caratteri di "newline" [nuova riga] vengono gestiti automaticamente dal Widget Label. È possibile utilizzare \n per creare un'interruzione di riga. Ad esempio: "line1\nline2\n\nline4"

### Modalità lunghe

Per default, la larghezza e l'altezza dell'etichetta sono impostate su `LV_SIZE_CONTENT`. In questo modo, la dimensione dell'etichetta viene automaticamente espansa in base alla dimensione del testo + padding + larghezza del bordo. In alternativa, se la larghezza o l'altezza vengono impostate esplicitamente (ad esempio, utilizzando `lv_obj_set_width()` o un layout), le righe più larghe della larghezza dell'etichetta possono essere manipolate in base a diverse policy di "modalità lunga". Analogamente, le policy possono essere applicate se l'altezza del testo è maggiore dell'altezza della Label.

- `LV_LABEL_LONG_WRAP` Manda a capo le righe troppo lunghe. Se l'altezza è `LV_SIZE_CONTENT`, l'altezza dell'etichetta verrà espansa, altrimenti il testo verrà tagliato. (Il default)
- `LV_LABEL_LONG_DOT` Sostituisce gli ultimi 3 caratteri dall'angolo in basso a destra dell'etichetta con punti (..)
- `LV_LABEL_LONG_SCROLL` Se il testo è più largo dell'etichetta, scorrerlo orizzontalmente avanti e indietro. Se è più alto, scorrere verticalmente. Lo scorrimento avviene in una sola direzione e lo scorrimento orizzontale ha la precedenza.
- `LV_LABEL_LONG_SCROLL_CIRCULAR` Se il testo è più largo dell'etichetta, scorrerlo orizzontalmente in modo continuo. Se è più alto, scorrere verticalmente. Lo scorrimento avviene in una sola direzione e lo scorrimento orizzontale ha la precedenza.
- `LV_LABEL_LONG_CLIP` Ritaglia semplicemente le parti di testo esterne all'etichetta.

È possibile specificare la modalità lunga con `lv_label_set_long_mode(label, LV_LABEL_LONG_...)`

Si noti che `LV_LABEL_LONG_DOT` manipola il buffer di testo sul posto per aggiungere/rimuovere i punti. Quando si utilizzano `lv_label_set_text()` o `lv_label_set_array_text()`, viene allocato un buffer separato e questo dettaglio di implementazione passa inosservato. Questo non avviene con `lv_label_set_text_static()`. Il buffer che passi a `lv_label_set_text_static()` deve essere scrivibile per utilizzare `LV_LABEL_LONG_DOT`.

### Ricolorazione del testo

Nel testo, è possibile utilizzare comandi per ricoloreare parti del testo. Ad esempio: Write a #ff0000 red# word. Questa funzionalità può essere abilitata individualmente per ogni etichetta tramite la funzione `lv_label_set_recolor(label, en)`. Nel contesto del testo con ritorno a capo automatico, qualsiasi ricoloreazione avviata su una riga verrà terminata alla fine della riga in cui la riga è andata a capo, se non era già terminata da un # finale nel testo.

## Selezione del testo

Se abilitata tramite `LV_LABEL_TEXT_SELECTION`, è possibile selezionare parte del testo. È simile a quando si utilizza il mouse su un PC per selezionare il testo. L'intero meccanismo (cliccare e selezionare il testo mentre si trascina il dito/mouse) è implementato in `Text Area (lv_textarea)` e il Widget Label consente solo la selezione programmatica del testo con `lv_label_get_text_selection_start(label, start_char_index)` e `lv_label_get_text_selection_end(label, end_char_index)`.

## Allineamento del testo

Per allineare orizzontalmente le righe di un'etichetta, la proprietà di stile `text_align` può essere utilizzata con `lv_obj_set_style_text_align()` o `lv_style_set_text_align()`, passando uno dei valori di enumerazione `LV_TEXT_ALIGN_...`. Si noti che questo ha un effetto visibile solo se:

- la larghezza del Widget Label è maggiore della larghezza della riga di testo più lunga e
- il testo ha più righe con lunghezze diverse.

## Testo molto lungo

LVGL può gestire in modo efficiente etichette molto lunghe (ad esempio > 40k caratteri) salvando alcuni dati extra (~12 byte) per velocizzare il disegno. Per abilitare questa funzionalità, impostare `LV_LABEL_LONG_TXT_HINT` a 1 in `lv_conf.h`.

## Animazioni di scorrimento personalizzate

Alcuni aspetti delle animazioni di scorrimento nelle modalità lunghe `LV_LABEL_LONG_SCROLL` e `LV_LABEL_LONG_SCROLL_CIRCULAR` possono essere personalizzati impostando la proprietà di stile dell'animazione dell'etichetta, utilizzando `lv_style_set_anim()`. Verrà trattato come un template che verrà utilizzato per creare le animazioni di scorrimento.

## Simboli

Le etichette possono visualizzare simboli accanto alle lettere (o da soli). Leggere la sezione `Font (lv_font)` per saperne di più sui simboli.

### 6.17.4 Eventi

I Widget Label non inviano eventi speciali. Per default, i Widget Label vengono creati senza il flag `LV_OBJ_FLAG_CLICKABLE`, ma è possibile aggiungerlo per far sì che un Widget Label emetta eventi `LV_EVENT_CLICKED`, se lo si desidera.

#### Ulteriori Informazioni

Scoprire di più sugli `Eventi del Widget di Base` emessi da tutti i Widget.

Scoprire di più sugli `Eventi`.

### 6.17.5 Tasti

Nessun `Tasto` viene elaborato dai Widget Label.

#### Ulteriori Informazioni

Scoprire di più sui `Tasti`.

## 6.17.6 Esempio

#### A capo automatico, ricolorazione e scorrimento

```
#include "../../lv_examples.h"
#ifndef LV_USE_LABEL && LV_BUILD_EXAMPLES

/***
 * Show line wrap, re-color, line align and text scrolling.
 */
void lv_example_label_1(void)
{
    lv_obj_t * label1 = lv_label_create(lv_screen_active());
    lv_label_set_long_mode(label1, LV_LABEL_LONG_MODE_WRAP); /*Break the long lines*/
    lv_label_set_recolor(label1, true); /*Enable re-coloring by commands in the text*/
    lv_label_set_text(label1, "#0000ff Re-color# #ff0000 words# #ff0000 of a# label, align the lines to the center "
                           "and wrap long text automatically.");
    lv_obj_set_width(label1, 150); /*Set smaller width to make the lines wrap*/
    lv_obj_set_style_text_align(label1, LV_TEXT_ALIGN_CENTER, 0);
    lv_obj_align(label1, LV_ALIGN_CENTER, 0, -40);

    lv_obj_t * label2 = lv_label_create(lv_screen_active());
    lv_label_set_long_mode(label2, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR); /*Circular scroll*/
    lv_obj_set_width(label2, 150);
    lv_label_set_text(label2, "It is a circularly scrolling text. ");
    lv_obj_align(label2, LV_ALIGN_CENTER, 0, 40);
}

#endif
```

## Ombreggiatura del testo

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/***
 * Create a fake text shadow
 */
void lv_example_label_2(void)
{
    /*Create a style for the shadow*/
    static lv_style_t style_shadow;
    lv_style_init(&style_shadow);
    lv_style_set_text_opa(&style_shadow, LV_OPA_30);
    lv_style_set_text_color(&style_shadow, lv_color_black());

    /*Create a label for the shadow first (it's in the background)*/
    lv_obj_t * shadow_label = lv_label_create(lv_screen_active());
    lv_obj_add_style(shadow_label, &style_shadow, 0);

    /*Create the main label*/
    lv_obj_t * main_label = lv_label_create(lv_screen_active());
    lv_label_set_text(main_label, "A simple method to create\n"
                           "shadows on a text.\n"
                           "It even works with\n"
                           "newlines      and spaces.");
}

/*Set the same text for the shadow label*/
lv_label_set_text(shadow_label, lv_label_get_text(main_label));

/*Position the main label*/
lv_obj_align(main_label, LV_ALIGN_CENTER, 0, 0);

/*Shift the second label down and to the right by 2 pixel*/
lv_obj_align_to(shadow_label, main_label, LV_ALIGN_TOP_LEFT, 2, 2
}

#endif
```

## Visualizzazione di testi LTR, RTL e in cinese

## Disegno di etichette con gradiente di colore

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_FONT_MONTserrat_24 && LV_USE_CANVAS && LV_BUILD_EXAMPLES && LV_DRAW_SW_COMPLEX

#define MASK_WIDTH 150
#define MASK_HEIGHT 60

static void generate_mask(lv_draw_buf_t * mask, int32_t w, int32_t h, const char * txt)
{
    /*Create a "8 bit alpha" canvas and clear it*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, mask);
    lv_canvas_fill_bg(canvas, lv_color_black(), LV_OPA_TRANSP);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    /*Draw a label to the canvas. The result "image" will be used as mask*/
    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_color_white();
    label_dsc.align = LV_TEXT_ALIGN_CENTER;
    label_dsc.text = txt;
    label_dsc.font = &lv_font_montserrat_24;
    lv_area_t a = {0, 0, w - 1, h - 1};
    lv_draw_label(&layer, &label_dsc, &a);

    lv_canvas_finish_layer(canvas, &layer);

    lv_obj_delete(canvas);
}

/**
 * Draw label with gradient color
 */
void lv_example_label_4(void)
{
    /* Create the mask of a text by drawing it to a canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(mask, MASK_WIDTH, MASK_HEIGHT, LV_COLOR_FORMAT_L8);
    LV_DRAW_BUF_INIT_STATIC(mask);

    generate_mask(&mask, MASK_WIDTH, MASK_HEIGHT, "Text with gradient");

    /* Create an object from where the text will be masked out.
     * Now it's a rectangle with a gradient but it could be an image too*/
    lv_obj_t * grad = lv_obj_create(lv_screen_active());
    lv_obj_set_size(grad, MASK_WIDTH, MASK_HEIGHT);
    lv_obj_center(grad);
    lv_obj_set_style_bg_color(grad, lv_color_hex(0xffff00), 0);
    lv_obj_set_style_bg_grad_color(grad, lv_color_hex(0x0000ff), 0);
    lv_obj_set_style_bg_grad_dir(grad, LV_GRAD_DIR_HOR, 0);
    lv_obj_set_style_bitmap_mask_src(grad, &mask, 0);
}
#endif
```

## Personalizzazione dell'animazione a scorrimento circolare

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES

/***
 * Show customizing the circular scrolling animation of a label with `LV_LABEL_LONG_MODE_SCROLL_CIRCULAR`
 * long mode.
 */
void lv_example_label_5(void)
{
    static lv_anim_t animation_template;
    static lv_style_t label_style;

    lv_anim_init(&animation_template);
    lv_anim_set_delay(&animation_template, 1000);           /*Wait 1 second to start the first scroll*/
    lv_anim_set_repeat_delay(&animation_template,
                           3000);           /*Repeat the scroll 3 seconds after the label scrolls back to the initial position*/
    lv_anim_set_repeat_count(&animation_template, LV_ANIM_REPEAT_INFINITE);

    /*Initialize the label style with the animation template*/
    lv_style_init(&label_style);
    lv_style_set_anim(&label_style, &animation_template);

    lv_obj_t * label1 = lv_label_create(lv_screen_active());
    lv_label_set_long_mode(label1, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);           /*Circular scroll*/
    lv_obj_set_width(label1, 150);
    lv_label_set_text(label1, "It is a circularly scrolling text. ");
    lv_obj_align(label1, LV_ALIGN_CENTER, 0, 40);
    lv_obj_add_style(label1, &label_style, LV_STATE_DEFAULT);           /*Add the style to the label*/
}

#endif
```

## Carattere a spaziatura fissa

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_BUILD_EXAMPLES && LV_FONT_MONTserrat_20

static bool fix_w_get_glyph_dsc(const lv_font_t * font, lv_font_glyph_dsc_t * dsc, uint32_t letter,
                                uint32_t letter_next)
{
    bool ret = lv_font_get_glyph_dsc_fmt_txt(font, dsc, letter, letter_next);
    if(!ret) return false;

    /* Set a fixed width */
    dsc->adv_w = 20;
```

(continues on next page)

(continua dalla pagina precedente)

```

dsc->ofs_x = (dsc->adv_w - dsc->box_w) / 2;
return true;
}

void lv_example_label_6(void)
{
    /* Clone the original font and override its behavior */
    static lv_font_t mono_font;
    mono_font = lv_font_montserrat_20;
    mono_font.get_glyph_dsc = fix_w_get_glyph_dsc;

    /* Create a label with normal font */
    lv_obj_t * label1 = lv_label_create(lv_screen_active());
    lv_obj_set_style_text_font(label1, &lv_font_montserrat_20, 0);
    lv_label_set_text(label1, "0123.Wabc");

    /* Create a label with fixed-width glyph descriptor override */
    lv_obj_t * label2 = lv_label_create(lv_screen_active());
    lv_obj_set_y(label2, 30);
    lv_obj_set_style_text_font(label2, &mono_font, 0);
    lv_label_set_text(label2, "0123.Wabc");
}

#endif

```

## 6.17.7 API

*lv\_observer.h*

*lv\_label.h*

*lv\_label\_private.h*

*lv\_obj\_property\_names.h*

## 6.18 LED (lv\_led)

### 6.18.1 Panoramica

I LED sono Widget rettangolari (o circolari) la cui luminosità può essere regolata. Con una luminosità inferiore, il colore del LED diventa più scuro.

### 6.18.2 Parti e Stili

- LV\_LED\_PART\_MAIN utilizza le *tipiche proprietà di stile del background*.

### 6.18.3 Utilizzo

#### Color

È possibile impostare il colore del LED con `lv_led_set_color(led, lv_color_hex(0xff0080))`. Questo verrà utilizzato come colore di sfondo, colore del bordo e colore dell'ombra.

#### Luminosità

È possibile impostare la luminosità con `lv_led_set_brightness(led, brightness)`. Il valore di `brightness` deve essere compreso tra 0 (più scuro) e 255 (più chiaro).

#### Attiva/disattiva

Si usano `lv_led_on(led)` e `lv_led_off(led)` per impostare la luminosità su un valore predefinito ON o OFF. `lv_led_toggle(led)` alterna tra lo stato ON e OFF.

È possibile impostare valori di luminosità LED ON e OFF personalizzati definendo le macro LV\_LED\_BRIGHT\_MAX e LV\_LED\_BRIGHT\_MIN nel progetto. I loro valori di default sono 80 e 255. Anche questi devono essere compresi nell'intervallo [0..255].

## 6.18.4 Eventi

I Widget LED non inviano eventi speciali.

### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

## 6.18.5 Tasti

I Widget LED non elaborano alcun *tasto*.

### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

## 6.18.6 Esempio

### LED con stile personalizzato

```
#include "../../lv_examples.h"
#if LV_USE_LED && LV_BUILD_EXAMPLES

/***
 * Create LED's with different brightness and color
 */
void lv_example_led_1(void)
{
    /*Create a LED and switch it OFF*/
    lv_obj_t * led1 = lv_led_create(lv_screen_active());
    lv_obj_align(led1, LV_ALIGN_CENTER, -80, 0);
    lv_led_off(led1);

    /*Copy the previous LED and set a brightness*/
    lv_obj_t * led2 = lv_led_create(lv_screen_active());
    lv_obj_align(led2, LV_ALIGN_CENTER, 0, 0);
    lv_led_set_brightness(led2, 150);
    lv_led_set_color(led2, lv_palette_main(LV_PALETTE_RED));

    /*Copy the previous LED and switch it ON*/
    lv_obj_t * led3 = lv_led_create(lv_screen_active());
    lv_obj_align(led3, LV_ALIGN_CENTER, 80, 0);
    lv_led_on(led3);
}

#endif
```

## 6.18.7 API

*lv\_led.h*

*lv\_led\_private.h*

## 6.19 Line (lv\_line)

### 6.19.1 Panoramica

Il widget "Line" è in grado di disegnare linee rette tra un insieme di punti.

### 6.19.2 Parti e Stili

- *LV\_PART\_MAIN* utilizza le proprietà *typical background* e lo stile della linea.

### 6.19.3 Utilizzo

#### Impostazione dei punti

I punti di una linea devono essere memorizzati in un array `lv_point_precise_t` e passati al Widget tramite la funzione `lv_line_set_points(lines, point_array, point_cnt)`.

Le loro coordinate possono essere specificate come coordinate pixel grezze (ad esempio `{5, 10}`) o come percentuale del "bounding box" [riquadro di delimitazione] della linea utilizzando `lv_pct(x)`. In quest'ultimo caso, potrebbe essere necessario impostare esplicitamente la larghezza/altezza della Linea utilizzando `lv_obj_set_width()` e `lv_obj_set_height()`, poiché i valori percentuali non espandono automaticamente il riquadro di delimitazione.

#### Dimensionamento automatico

Per default, la larghezza e l'altezza della Linea sono impostate su `LV_SIZE_CONTENT`. Ciò significa che le sue dimensioni verranno impostate automaticamente in modo da adattarsi a tutti i punti. Se le dimensioni vengono impostate esplicitamente, alcune parti della Linea potrebbero non essere visibili.

#### Inversione della y

Per default, il punto `y == 0` si trova nella parte superiore del Widget. Potrebbe essere contortintuitivo in alcuni casi, quindi le coordinate y possono essere invertite con `lv_line_set_y_invert(line, true)`. In questo caso, `y == 0` sarà in fondo al Widget. L'inversione della y è disabilitata per default.

### 6.19.4 Eventi

I Widget Line inviano solo *eventi generici*.

#### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

### 6.19.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Line.

#### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

### 6.19.6 Esempio

#### Linea semplice

```
#include "../../lv_examples.h"
#if LV_USE_LINE && LV_BUILD_EXAMPLES

void lv_example_line_1(void)
{
    /*Create an array for the points of the line*/
    static lv_point_precise_t line_points[] = { {5, 5}, {70, 70}, {120, 10}, {180, 60}, {240, 10} };

    /*Create style*/
    static lv_style_t style_line;
    lv_style_init(&style_line);
    lv_style_set_line_width(&style_line, 8);
    lv_style_set_line_color(&style_line, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_line_rounded(&style_line, true);

    /*Create a line and apply the new style*/
    lv_obj_t * line1;
    line1 = lv_line_create(lv_screen_active());
    lv_line_set_points(line1, line_points, 5);      /*Set the points*/
    lv_obj_add_style(line1, &style_line, 0);
    lv_obj_center(line1);
}
```

(continues on next page)

(continua dalla pagina precedente)

```
}
```

```
#endif
```

## 6.19.7 API

*lv\_line.h*

*lv\_line\_private.h*

# 6.20 List (lv\_list)

## 6.20.1 Panoramica

Il widget List è fondamentalmente un rettangolo con layout verticale a cui è possibile aggiungere Pulsanti e Testo.

## 6.20.2 Parti e Stili

### Background

- **LV\_PART\_MAIN** La parte principale dell'elenco che utilizza le *tipiche proprietà di stile del background*.
- **LV\_PART\_SCROLLBAR** La scrollbar. Consultare la documentazione di *Widget Base (lv\_obj)* per i dettagli.

### Pulsanti e Testo

- Consultare la documentazione del *Button*'s e della *Label*.

## 6.20.3 Utilizzo

### Pulsanti

*lv\_list\_add\_button(list, icon, text)* aggiunge un pulsante a larghezza intera con un'icona (che può essere un'immagine o un simbolo) e del testo. Questa funzione restituisce un puntatore al pulsante creato, utilizzabile, ad esempio, per aggiungere un callback di evento.

Il testo scorre orizzontalmente se è più lungo del pulsante.

### Text

*lv\_list\_add\_text(list, text)* aggiunge una stringa di testo. Questa funzione restituisce un puntatore all'etichetta creata, utilizzabile, ad esempio, per modificarne il testo con una delle funzioni *lv\_label\_set\_text...()*.

## 6.20.4 Eventi

I Widget List non inviano eventi speciali, ma gli eventi possono essere inviati dai Pulsanti come di consueto.

### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

## 6.20.5 Tasti

Nessun *Tasto* viene elaborato dai Widget List.

### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

## 6.20.6 Esempio

### Lista semplice

```
#include "../../../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES
static lv_obj_t * list1;

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        LV_UNUSED(obj);
        LV_LOG_USER("Clicked: %s", lv_list_get_button_text(list1, obj));
    }
}
void lv_example_list_1(void)
{
    /*Create a list*/
    list1 = lv_list_create(lv_screen_active());
    lv_obj_set_size(list1, 180, 220);
    lv_obj_center(list1);

    /*Add buttons to the list*/
    lv_obj_t * btn;
    lv_list_add_text(list1, "File");
    btn = lv_list_add_button(list1, LV_SYMBOL_FILE, "New");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_DIRECTORY, "Open");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_SAVE, "Save");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_CLOSE, "Delete");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_EDIT, "Edit");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Connectivity");
    btn = lv_list_add_button(list1, LV_SYMBOL_BLUETOOTH, "Bluetooth");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_GPS, "Navigation");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_USB, "USB");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_BATTERY_FULL, "Battery");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_list_add_text(list1, "Exit");
    btn = lv_list_add_button(list1, LV_SYMBOL_OK, "Apply");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
    btn = lv_list_add_button(list1, LV_SYMBOL_CLOSE, "Close");
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);
}
#endif
```

### Ordinamento di un elenco utilizzando i pulsanti Su e Giù

```
#include "../../../lv_examples.h"
#if LV_USE_LIST && LV_BUILD_EXAMPLES
static lv_obj_t * list1;
static lv_obj_t * list2;

static lv_obj_t * currentButton = NULL;

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED) {
        LV_LOG_USER("Clicked: %s", lv_list_get_button_text(list1, obj));

        if(currentButton == obj) {
            currentButton = NULL;
        } else {
            currentButton = obj;
        }
        lv_obj_t * parent = lv_obj_get_parent(obj);
        int32_t i;
        for(i = 0; i < (int32_t)lv_obj_get_child_count(parent); i++) {
            lv_obj_t * child = lv_obj_get_child(parent, i);
            if(child == currentButton) {
                lv_obj_add_state(child, LV_STATE_CHECKED);
            } else {
                lv_obj_remove_state(child, LV_STATE_CHECKED);
            }
        }
    }
}

static void event_handler_top(lv_event_t * e)
{
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_event_code_t code = lv_event_get_code(e);
if(code == LV_EVENT_CLICKED) {
    if(currentButton == NULL) return;
    lv_obj_move_background(currentButton);
    lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
}
}

static void event_handler_up(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;
        int32_t index = lv_obj_get_index(currentButton);
        if(index <= 0) return;
        lv_obj_move_to_index(currentButton, index - 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_center(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;
        lv_obj_t * parent = lv_obj_get_parent(currentButton);
        const int32_t pos = (int32_t)lv_obj_get_child_count(parent) / 2;
        lv_obj_move_to_index(currentButton, pos);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_dn(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        if(currentButton == NULL) return;
        const int32_t index = lv_obj_get_index(currentButton);

        lv_obj_move_to_index(currentButton, index + 1);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_bottom(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_CLICKED) {
        if(currentButton == NULL) return;
        lv_obj_move_foreground(currentButton);
        lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
    }
}

static void event_handler_swap(lv_event_t * e)
{
    const lv_event_code_t code = lv_event_get_code(e);
    if((code == LV_EVENT_CLICKED) || (code == LV_EVENT_LONG_PRESSED_REPEAT)) {
        uint32_t cnt = lv_obj_get_child_count(list1);
        for(int i = 0; i < 100; i++) {
            lv_obj_t * obj = lv_obj_get_child(list1, (int32_t)lv_rand(0, cnt - 1));
            lv_obj_move_to_index(obj, (int32_t)lv_rand(0, cnt - 1));
            if(currentButton != NULL) {
                lv_obj_scroll_to_view(currentButton, LV_ANIM_ON);
            }
        }
    }
}

void lv_example_list_2(void)
{
    /*Create a list*/
    list1 = lv_list_create(lv_screen_active());
    lv_obj_set_size(list1, lv_pct(60), lv_pct(100));
    lv_obj_set_style_pad_row(list1, 5, 0);

    /*Add buttons to the list*/
    lv_obj_t * btn;
    int i;
    for(i = 0; i < 15; i++) {
        btn = lv_button_create(list1);
        lv_obj_set_width(btn, lv_pct(50));
        lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

        lv_obj_t * lab = lv_label_create(btn);
        lv_label_set_text_fmt(lab, "Item %d", i);
    }

    /*Select the first button by default*/
    currentButton = lv_obj_get_child(list1, 0);
    lv_obj_add_state(currentButton, LV_STATE_CHECKED);

    /*Create a second list with up and down buttons*/
    list2 = lv_list_create(lv_screen_active());
    lv_obj_set_size(list2, lv_pct(40), lv_pct(100));
    lv_obj_align(list2, LV_ALIGN_TOP_RIGHT, 0, 0);
    lv_obj_set_flex_flow(list2, LV_FLEX_FLOW_COLUMN);

    btn = lv_list_add_button(list2, NULL, "Top");
    lv_obj_add_event_cb(btn, event_handler_top, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_button(list2, LV_SYMBOL_UP, "Up");
    lv_obj_add_event_cb(btn, event_handler_up, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);

    btn = lv_list_add_button(list2, LV_SYMBOL_LEFT, "Center");
    lv_obj_add_event_cb(btn, event_handler_center, LV_EVENT_ALL, NULL);
    lv_group_remove_obj(btn);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

btn = lv_list_add_button(list2, LV_SYMBOL_DOWN, "Down");
lv_obj_add_event_cb(btn, event_handler_dn, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, NULL, "Bottom");
lv_obj_add_event_cb(btn, event_handler_bottom, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);

btn = lv_list_add_button(list2, LV_SYMBOL_SHUFFLE, "Shuffle");
lv_obj_add_event_cb(btn, event_handler_swap, LV_EVENT_ALL, NULL);
lv_group_remove_obj(btn);
}

#endif

```

## 6.20.7 API

*lv\_api\_map\_v8.h*

*lv\_list.h*

# 6.21 Lottie (*lv\_lottie*)

## 6.21.1 Panoramica

Il Widget Lottie è in grado di analizzare, rasterizzare e riprodurre le animazioni Lottie.

Le animazioni Lottie sono animazioni vettoriali. Le si consideri una moderna combinazione di SVG e GIF.

Le animazioni possono essere scaricate da diverse fonti, come <https://lottiefiles.com/> oppure se ne possono creare di proprie, ad esempio, utilizzando Adobe After Effects.

Il Widget Lottie si basa su *Canvas* (*lv\_canvas*) perché per eseguire il rendering dell'animazione l'utente deve fornire un buffer in cui memorizzare il fotogramma corrente.

## 6.21.2 Parti e Stili

- *LV\_PART\_MAIN* Il background dell'animazione Lottie. Si applicano le tipiche proprietà di stile del background, ma di solito viene lasciato trasparente.

## 6.21.3 Utilizzo

### Dipendenze

Il Widget Lottie utilizza la libreria ThorVG che è integrata in LVGL. Per utilizzare le animazioni Lottie, è necessario abilitare *LV\_USE\_THORVG\_INTERNAL* (per utilizzare il ThorVG integrato) o *LV\_USE\_THORVG\_EXTERNAL* (per linkarlo esternamente) in *lv\_conf.h*. Per la grafica vettoriale in generale, è necessario abilitare anche *LV\_USE\_VECTOR\_GRAPHIC*.

Poiché ThorVG è scritto in C++, quando si utilizza *LV\_USE\_THORVG\_INTERNAL* assicurarsi di poter compilare i file cpp.

### Impostare un buffer

Per eseguire il rendering dell'animazione, è necessario assegnare un buffer al Widget Lottie. Le animazioni vengono renderizzate in formato ARGB8888, pertanto la dimensione del buffer deve essere pari a *target\_width* x *target\_height* x 4 byte.

Per mantenere coerenti le dimensioni del buffer e quelle dell'animazione, la dimensione del Widget (ovvero la dimensione dell'animazione) viene impostata internamente sulle dimensioni del buffer.

Il buffer può essere impostato con *lv\_lottie\_set\_buffer(lottie, w, h, buf)* o *lv\_lottie\_set\_draw\_buf(lottie, draw\_buf)*.

Quando si utilizza un buffer di disegno, questo deve essere già inizializzato dall'utente con il formato colore *LV\_COLOR\_FORMAT\_ARGB8888\_PREMULTIPLIED*.

## Impostare una sorgente

`lv_example_lottie_approve.c` contiene un'animazione di esempio. Invece di memorizzare la stringa JSON, viene memorizzato un array esadecimale per i seguenti motivi:

- per evitare l'escape del carattere " nel file JSON e
- alcuni compilatori non supportano stringhe molto lunghe.

`lvgl/scripts/filetohex.py` può essere utilizzato per convertire un file Lottie in un array esadecimale. Ad esempio:

```
./filetohex.py path/to/lottie.json --filter-character --null-terminate > out.txt
```

`--filter-character` filtra i caratteri non ASCII e `--null-terminate` assicura che venga aggiunto uno zero finale per chiudere correttamente la stringa.

Per creare un'animazione dai dati, si usa `lv_lottie_set_src_data(lottie, data, sizeof(data))`

Le animazioni Lottie possono essere aperte da file JSON utilizzando `lv_lottie_set_src_file(lottie, "path/to/file.json")`. Notare che il caricatore Lottie non supporta l'interfaccia del file system di LVGL, ma dovrebbe essere utilizzato un "path normale" senza lettera del driver.

## Ottenere l'animazione

```
lv_anim_t * a = lv_lottie_get_anim(lottie)
```

restituisce l'animazione LVGL che controlla l'animazione di Lottie. Per default, l'animazione gira all'infinito a 60 FPS, tuttavia l'animazione LVGL può essere regolata liberamente.

### 6.21.4 Eventi

Nessun evento viene emesso dai Widget di Lottie.

#### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

### 6.21.5 Tasti

Nessun tasto viene elaborato dai Widget Lottie.

#### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

### 6.21.6 Esempio

#### Caricamento di un'animazione Lottie da un array

```
#include "../../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES
#define LV_USE_LOTTIE

/***
 * Load an lottie animation from data
 */
void lv_example_lottie_1(void)
{
    extern const uint8_t lv_example_lottie_approve[];
    extern const size_t lv_example_lottie_approve_size;

    lv_obj_t * lottie = lv_lottie_create(lv_screen_active());
    lv_lottie_set_src_data(lottie, lv_example_lottie_approve, lv_example_lottie_approve_size);
}
```

(continues on next page)

(continua dalla pagina precedente)

```
#if LV_DRAW_BUF_ALIGN == 4 && LV_DRAW_BUF_STRIDE_ALIGN == 1
    /*If there are no special requirements, just declare a buffer
     *x4 because the Lottie is rendered in ARGB8888_PREMULTIPLIED format*/
    static uint8_t buf[64 * 64 * 4];
    lv_lottie_set_buffer(lottie, 64, 64, buf);
#else
    /*For GPUs and special alignment/stride setting use a draw_buf instead*/
    LV_DRAW_BUF_DEFINE(draw_buf, 64, 64, LV_COLOR_FORMAT_ARGB8888_PREMULTIPLIED);
    lv_lottie_set_draw_buf(lottie, &draw_buf);
#endif

    lv_obj_center(lottie);
}

#else

void lv_example_lottie_1(void)
{
    /*fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Lottie cannot be previewed online");
    lv_obj_center(label);
}

#endif /*LV_USE_LOTTIE*/
#endif /*LV_BUILD_EXAMPLES/
```

## Caricamento di un'animazione Lottie da un file

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_LOTTIE

/***
 * Load an lottie animation from file
 */
void lv_example_lottie_2(void)
{
    lv_obj_t * lottie = lv_lottie_create(lv_screen_active());
    lv_lottie_set_src_file(lottie, "lvgl/examples/widgets/lottie/lv_example_lottie_approve.json");

#if LV_DRAW_BUF_ALIGN == 4 && LV_DRAW_BUF_STRIDE_ALIGN == 1
    /*If there are no special requirements, just declare a buffer
     *x4 because the Lottie is rendered in ARGB8888_PREMULTIPLIED format*/
    static uint8_t buf[64 * 64 * 4];
    lv_lottie_set_buffer(lottie, 64, 64, buf);
#else
    /*For GPUs and special alignment/stride setting use a draw_buf instead*/
    LV_DRAW_BUF_DEFINE(draw_buf, 64, 64, LV_COLOR_FORMAT_ARGB8888_PREMULTIPLIED);
    lv_lottie_set_draw_buf(lottie, &draw_buf);
#endif

    lv_obj_center(lottie);
}

#else

void lv_example_lottie_2(void)
{
    /*fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Lottie cannot be previewed online");
    lv_obj_center(label);
}

#endif /*LV_USE_LOTTIE*/
#endif /*LV_BUILD_EXAMPLES/
```

## 6.21.7 API

*lv\_lottie.h*

*lv\_lottie\_private.h*

## 6.22 Menu (lv\_menu)

### 6.22.1 Panoramica

Il Widget Menu può essere utilizzato per creare menù multilivello che gestiscono automaticamente la navigazione tra i livelli di menù e consentono all'utente di catturare la navigazione tra le pagine e gli eventi di clic.

## 6.22.2 Parti e Stili

Il Widget Menu è costituito dai seguenti Widget:

- Contenitore principale: `lv_menu_main_cont`
- Intestazione principale: `lv_menu_main_header_cont`
- Pulsante Back: *Button (lv\_button)*
- Icona pulsante Back: *Image (lv\_image)*
- Pagina Principale: `lv_menu_page`
- Contenitore barra laterale: `lv_menu_sidebar_cont`
- Intestazione barra laterale: `lv_menu_sidebar_header_cont`
- Pulsante Back: *Button (lv\_button)*
- Icona pulsante Back: *Image (lv\_image)*
- Pagina Barra Laterale: `lv_menu_page`

## 6.22.3 Utilizzo

### Creare un Menu

`lv_menu_create(parent)` crea un nuovo Menu vuoto.

Questo crea un Widget Menu con questa gerarchia di oggetti:

- Menu: `lv_menu_t`
  - Hidden Sub-Page Storage: `lv_obj_t`
  - Contenitore principale: `lv_menu_main_cont_class`
    - \* Intestazione principale: `lv_menu_main_header_cont_class`
      - Pulsante Back: *Button (lv\_button)*
      - Icona pulsante Back: *Image (lv\_image)*
      - Titolo dell'intestazione principale: *Label (lv\_label)* (default hidden)

Nota: non viene creata alcuna barra laterale. Se ne può creare una in seguito, volendo.

### Modalità intestazione

Esistono le seguenti modalità di intestazione:

- `LV_MENU_HEADER_TOP_FIXED` L'intestazione è posizionata in alto. (il default)
- `LV_MENU_HEADER_TOP_UNFIXED` L'intestazione è posizionata in alto e può essere fatta scorrere fuori dalla vista.
- `LV_MENU_HEADER_BOTTOM_FIXED` L'intestazione è posizionata in basso.

È possibile impostare le modalità dell'intestazione con `lv_menu_set_mode_header(menu, LV_MENU_HEADER...)`.

### Modalità del pulsante Root back

Esistono le seguenti modalità del pulsante "root back":

- `LV_MENU_ROOT_BACK_BTN_DISABLED` (il default)
- `LV_MENU_ROOT_BACK_BTN_ENABLED`

È possibile impostare le modalità del pulsante "root back" con `lv_menu_set_mode_root_back_button(menu, LV_MENU_ROOT_BACK_BTN...)`.

## Creare un Menu Page

`lv_menu_page_create(menu, title)` crea un nuovo Menu Page vuoto. È possibile aggiungere qualsiasi Widget alla Page.

## Impostare una Menu Page nell'area principale

Una volta creata una Menu Page, è possibile impostarla nell'area principale con `lv_menu_set_page(menu, page)`. NULL per cancellare il main e la cronologia del Menu.

## Impostare una Menu Page nella barra laterale

Una volta creata una Menu Page, la si può impostare sulla barra laterale con `lv_menu_set_sidebar_page(menu, page)`. NULL per cancellare la barra laterale.

## Collegamento tra Menu Page

Se, ad esempio, è stato creato un oggetto button nella Page principale. Quando si clicca sul Widget pulsante, si vuole che si apra una nuova Page, si usa `lv_menu_set_load_page_event(menu, btn, new page)`.

## Crea un contenitore, una sezione e un separatore del Menu

È possibile creare i seguenti oggetti per semplificare la definizione dello stile del Menu:

- `lv_menu_cont_create(parent_page)` crea un nuovo contenitore vuoto.
- `lv_menu_section_create(parent_page)` crea una nuova sezione vuota.
- `lv_menu_separator_create(parent_page)` crea un separatore.

## 6.22.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando viene visualizzata una Page.
  - `lv_menu_get_cur_main_page(menu)` restituisce un puntatore alla Menu Page attualmente visualizzata nel contenitore principale.
  - `lv_menu_get_cur_sidebar_page(menu)` restituisce un puntatore alla Menu Page attualmente visualizzata nel contenitore della barra laterale.
- `LV_EVENT_CLICKED` Inviato quando si clicca su un "back button" in un'intestazione dalla barra principale o laterale. `LV_OBJ_FLAG_EVENT_BUBBLE` è abilitato sui pulsanti, in modo da poter aggiungere eventi al Menu stesso.
  - `lv_menu_back_button_is_root(menu, button)` per verificare se il pulsante è il "root back button".

### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

## 6.22.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Menu.

### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

## 6.22.6 Esempio

### Menù semplice

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_BUILD_EXAMPLES

void lv_example_menu_1(void)
{
    /*Create a menu object*/
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);
}

#endif
```

### Menù semplice con pulsante principale

```
#include "../../lv_examples.h"
#if LV_USE_MENU && LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void back_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);

    if(lv_menu_back_button_is_root(menu, obj)) {
        lv_obj_t * mbox1 = lv_msgbox_create(NULL);
        lv_msgbox_add_title(mbox1, "Hello");
        lv_msgbox_add_text(mbox1, "Root back btn click.");
        lv_msgbox_add_close_button(mbox1);
    }
}

void lv_example_menu_2(void)
{
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_menu_set_mode_root_back_button(menu, LV_MENU_ROOT_BACK_BUTTON_ENABLED);
    lv_obj_add_event_cb(menu, back_event_handler, LV_EVENT_CLICKED, menu);
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2");

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_menu_set_page(menu, main_page);
}

#endif
```

## Menù semplice con intestazione personalizzata

```
#include "../../lv_examples.h"
#ifndef LV_USE_MENU && LV_BUILD_EXAMPLES

void lv_example_menu_3(void)
{
    /*Create a menu object*/
    lv_obj_t * menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    /*Modify the header*/
    lv_obj_t * back_btn = lv_menu_get_main_header_back_button(menu);
    lv_obj_t * back_button_label = lv_label_create(back_btn);
    lv_label_set_text(back_button_label, "Back");

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create sub pages*/
    lv_obj_t * sub_1_page = lv_menu_page_create(menu, "Page 1");
    cont = lv_menu_cont_create(sub_1_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    lv_obj_t * sub_2_page = lv_menu_page_create(menu, "Page 2");
    cont = lv_menu_cont_create(sub_2_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    lv_obj_t * sub_3_page = lv_menu_page_create(menu, "Page 3");
    cont = lv_menu_cont_create(sub_3_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Hello, I am hiding here");

    /*Create a main page*/
    lv_obj_t * main_page = lv_menu_page_create(menu, NULL);
    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 1 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_1_page);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 2 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_2_page);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text(label, "Item 3 (Click me!)");
    lv_menu_set_load_page_event(menu, cont, sub_3_page);

    lv_menu_set_page(menu, main_page);
}

#endif
```

## Menù semplice con pulsante mobile per aggiungere una nuova pagina di menu

```
#include "../../lv_examples.h"
#ifndef LV_USE_MENU && LV_BUILD_EXAMPLES

static uint32_t btn_cnt = 1;
static lv_obj_t * main_page;
static lv_obj_t * menu;

static void float_button_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);

    btn_cnt++;

    lv_obj_t * cont;
    lv_obj_t * label;

    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);
    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
    lv_label_set_text_fmt(label, "Hello, I am hiding inside %" LV_PRIu32 "", btn_cnt);

    cont = lv_menu_cont_create(main_page);
    label = lv_label_create(cont);
    lv_label_set_text_fmt(label, "Item %" LV_PRIu32 "", btn_cnt);
    lv_menu_set_load_page_event(menu, cont, sub_page);

    lv_obj_scroll_to_view_recursive(cont, LV_ANIM_ON);
}

void lv_example_menu_4(void)
{
    /*Create a menu object*/
    menu = lv_menu_create(lv_screen_active());
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * label;

    /*Create a sub page*/
    lv_obj_t * sub_page = lv_menu_page_create(menu, NULL);
    cont = lv_menu_cont_create(sub_page);
    label = lv_label_create(cont);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_label_set_text(label, "Hello, I am hiding inside the first item");

/*Create a main page*/
main_page = lv_menu_page_create(menu, NULL);

cont = lv_menu_cont_create(main_page);
label = lv_label_create(cont);
lv_label_set_text(label, "Item 1");
lv_menu_set_load_page_event(menu, cont, sub_page);

lv_menu_set_page(menu, main_page);

/*Create floating btn*/
lv_obj_t * float_btn = lv_button_create(lv_screen_active());
lv_obj_set_size(float_btn, 50, 50);
lv_obj_add_flag(float_btn, LV_OBJ_FLAG_FLOATING);
lv_obj_align(float_btn, LV_ALIGN_BOTTOM_RIGHT, -10, -10);
lv_obj_add_event_cb(float_btn, float_button_event_cb, LV_EVENT_CLICKED, menu);
lv_obj_set_style_radius(float_btn, LV_RADIUS_CIRCLE, 0);
lv_obj_set_style_bg_image_src(float_btn, LV_SYMBOL_PLUS, 0);
lv_obj_set_style_text_font(float_btn, lv_theme_get_font_large(float_btn), 0);
}

#endif

```

## Menù complesso

```

#include "../../lv_examples.h"
#if LV_USE_MENU && LV_USE_MSGBOX && LV_BUILD_EXAMPLES

typedef enum {
    LV_MENU_ITEM_BUILDER_VARIANT_1,
    LV_MENU_ITEM_BUILDER_VARIANT_2
} lv_menu_builder_variant_t;

static void back_event_handler(lv_event_t * e);
static void switch_handler(lv_event_t * e);
static lv_obj_t * root_page;
static lv_obj_t * create_text(lv_obj_t * parent, const char * icon, const char * txt,
                             lv_menu_builder_variant_t builder_variant);
static lv_obj_t * create_slider(lv_obj_t * parent,
                               const char * icon, const char * txt, int32_t min, int32_t max, int32_t val);
static lv_obj_t * create_switch(lv_obj_t * parent,
                               const char * icon, const char * txt, bool chk);

void lv_example_menu_5(void)
{
    lv_obj_t * menu = lv_menu_create(lv_screen_active());

    lv_color_t bg_color = lv_obj_get_style_bg_color(menu, LV_PART_MAIN);
    if(lv_color_brightness(bg_color) > 127) {
        lv_obj_set_style_bg_color(menu, lv_color_darken(lv_obj_get_style_bg_color(menu, LV_PART_MAIN), 10), 0);
    } else {
        lv_obj_set_style_bg_color(menu, lv_color_darken(lv_obj_get_style_bg_color(menu, LV_PART_MAIN), 50), 0);
    }
    lv_menu_set_mode_root_back_button(menu, LV_MENU_ROOT_BACK_BUTTON_ENABLED);
    lv_obj_add_event_cb(menu, back_event_handler, LV_EVENT_CLICKED, menu);
    lv_obj_set_size(menu, lv_display_get_horizontal_resolution(NULL), lv_display_get_vertical_resolution(NULL));
    lv_obj_center(menu);

    lv_obj_t * cont;
    lv_obj_t * section;

    /*Create sub pages*/
    lv_obj_t * sub_mechanics_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_mechanics_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    lv_menu_separator_create(sub_mechanics_page);
    section = lv_menu_section_create(sub_mechanics_page);
    create_slider(section, LV_SYMBOL_SETTINGS, "Velocity", 0, 150, 120);
    create_slider(section, LV_SYMBOL_SETTINGS, "Acceleration", 0, 150, 50);
    create_slider(section, LV_SYMBOL_SETTINGS, "Weight limit", 0, 150, 80);

    lv_obj_t * sub_sound_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_sound_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    lv_menu_separator_create(sub_sound_page);
    section = lv_menu_section_create(sub_sound_page);
    create_switch(section, LV_SYMBOL_AUDIO, "Sound", false);

    lv_obj_t * sub_display_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_display_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    lv_menu_separator_create(sub_display_page);
    section = lv_menu_section_create(sub_display_page);
    create_slider(section, LV_SYMBOL_SETTINGS, "Brightness", 0, 150, 100);

    lv_obj_t * sub_software_info_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_software_info_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    section = lv_menu_section_create(sub_software_info_page);
    create_text(section, NULL, "Version 1.0", LV_MENU_ITEM_BUILDER_VARIANT_1);

    lv_obj_t * sub_legal_info_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_legal_info_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    section = lv_menu_section_create(sub_legal_info_page);
    for(uint32_t i = 0; i < 15; i++) {
        create_text(section, NULL,
                   "This is a long long long long long long long text, if it is long enough it may scroll.", LV_MENU_ITEM_BUILDER_VARIANT_1);
    }

    lv_obj_t * sub_about_page = lv_menu_page_create(menu, NULL);
    lv_obj_set_style_pad_hor(sub_about_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
    lv_menu_separator_create(sub_about_page);
    section = lv_menu_section_create(sub_about_page);
    cont = create_text(section, NULL, "Software information", LV_MENU_ITEM_BUILDER_VARIANT_1);
    lv_menu_set_load_page_event(menu, cont, sub_software_info_page);
    cont = create_text(section, NULL, "Legal information", LV_MENU_ITEM_BUILDER_VARIANT_1);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_menu_set_load_page_event(menu, cont, sub_legal_info_page);

lv_obj_t * sub_menu_mode_page = lv_menu_page_create(menu, NULL);
lv_obj_set_style_pad_hor(sub_menu_mode_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
lv_menu_separator_create(sub_menu_mode_page);
section = lv_menu_section_create(sub_menu_mode_page);
cont = create_switch(section, LV_SYMBOL_AUDIO, "Sidebar enable", true);
lv_obj_add_event_cb(lv_obj_get_child(cont, 2), switch_handler, LV_EVENT_VALUE_CHANGED, menu);

/*Create a root page*/
root_page = lv_menu_page_create(menu, "Settings");
lv_obj_set_style_pad_hor(root_page, lv_obj_get_style_pad_left(lv_menu_get_main_header(menu), LV_PART_MAIN), 0);
section = lv_menu_section_create(root_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Mechanics", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_mechanics_page);
cont = create_text(section, LV_SYMBOL_AUDIO, "Sound", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_sound_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Display", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_display_page);

create_text(root_page, NULL, "Others", LV_MENU_ITEM_BUILDER_VARIANT_1);
section = lv_menu_section_create(root_page);
cont = create_text(section, NULL, "About", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_about_page);
cont = create_text(section, LV_SYMBOL_SETTINGS, "Menu mode", LV_MENU_ITEM_BUILDER_VARIANT_1);
lv_menu_set_load_page_event(menu, cont, sub_menu_mode_page);

lv_menu_set_sidebar_page(menu, root_page);

lv_obj_send_event(lv_obj_get_child(lv_obj_get_child(lv_menu_get_cur_sidebar_page(menu), 0), 0), LV_EVENT_CLICKED,
NULL);
}

static void back_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);

    if(lv_menu_back_button_is_root(menu, obj)) {
        lv_obj_t * mbox1 = lv_msgbox_create(NULL);
        lv_msgbox.add_title(mbox1, "Hello");
        lv_msgbox.add_text(mbox1, "Root back btn click.");
        lv_msgbox.add_close_button(mbox1);
    }
}

static void switch_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * menu = (lv_obj_t *)lv_event_get_user_data(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        if(lv_obj_has_state(obj, LV_STATE_CHECKED)) {
            lv_menu_set_page(menu, NULL);
            lv_menu_set_sidebar_page(menu, root_page);
            lv_obj_send_event(lv_obj_get_child(lv_obj_get_child(lv_menu_get_cur_sidebar_page(menu), 0), 0), LV_EVENT_CLICKED,
NULL);
        }
        else {
            lv_menu_set_sidebar_page(menu, NULL);
            lv_menu_clear_history(menu); /* Clear history because we will be showing the root page later */
            lv_menu_set_page(menu, root_page);
        }
    }
}

static lv_obj_t * create_text(lv_obj_t * parent, const char * icon, const char * txt,
                           lv_menu_builder_variant_t builder_variant)
{
    lv_obj_t * obj = lv_menu_cont_create(parent);

    lv_obj_t * img = NULL;
    lv_obj_t * label = NULL;

    if(icon) {
        img = lv_image_create(obj);
        lv_image_set_src(img, icon);
    }

    if(txt) {
        label = lv_label_create(obj);
        lv_label_set_text(label, txt);
        lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
        lv_obj_set_flex_grow(label, 1);
    }

    if(builder_variant == LV_MENU_ITEM_BUILDER_VARIANT_2 && icon && txt) {
        lv_obj_add_flag(img, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
        lv_obj_swap(img, label);
    }

    return obj;
}

static lv_obj_t * create_slider(lv_obj_t * parent, const char * icon, const char * txt, int32_t min, int32_t max,
                             int32_t val)
{
    lv_obj_t * obj = create_text(parent, icon, txt, LV_MENU_ITEM_BUILDER_VARIANT_2);

    lv_obj_t * slider = lv_slider_create(obj);
    lv_obj_set_flex_grow(slider, 1);
    lv_slider_set_range(slider, min, max);
    lv_slider_set_value(slider, val, LV_ANIM_OFF);

    if(icon == NULL) {
        lv_obj_add_flag(slider, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
    }

    return obj;
}

static lv_obj_t * create_switch(lv_obj_t * parent, const char * icon, const char * txt, bool chk)

```

(continues on next page)

(continua dalla pagina precedente)

```
{
    lv_obj_t * obj = create_text(parent, icon, txt, LV_MENU_ITEM_BUILDER_VARIANT_1);
    lv_obj_t * sw = lv_switch_create(obj);
    lv_obj_add_state(sw, chk ? LV_STATE_CHECKED : LV_STATE_DEFAULT);
    return obj;
}
#endif
```

## 6.22.7 API

*lv\_menu.h*

*lv\_menu\_private.h*

# 6.23 Message Box (*lv\_msgbox*)

## 6.23.1 Panoramica

I "Message box" [finestre di messaggio] fungono da pop-up. Sono costituite da un'area di contenuto con un helper per l'aggiunta di testo, un'intestazione facoltativa (che può contenere un titolo, un pulsante di chiusura e altri pulsanti) e un piè di pagina facoltativo con pulsanti.

Il testo verrà suddiviso in più righe e l'altezza verrà impostata automaticamente. Se l'altezza viene impostata a livello di codice, il contenuto diventerà scorrevole.

La finestra di messaggio può essere modale (bloccando i clic sul resto dello schermo) o non modale.

## 6.23.2 Parti e Stili

La finestra di messaggio è costruita a partire da altri widget, quindi si può consultare la documentazione di questi Widget per i dettagli.

- Contenuto, intestazione e piè di pagina: *Widget Base (lv\_obj)*
- Pulsanti: *Button (lv\_button)*
- Titolo e testo del contenuto: *Label (lv\_label)*

## 6.23.3 Utilizzo

### Crea una message box

*lv\_msgbox\_create(parent)* crea una message box. Se **parent** è **NULL**, la finestra di messaggio sarà modale e utilizzerà il *Layer* del *Display di Default* come genitore.

### Aggiunta di pulsanti

Per aggiungere un pulsante [OK], [Cancel] o altri pulsanti per consentire all'utente di scegliere tra diverse risposte, aggiungere ciascun pulsante utilizzando la funzione *lv\_msgbox\_add\_footer\_button (msgbox, btn\_text)*. La chiamata a questa funzione aggiunge un piè di pagina (contenitore) se non ne è già presente uno e restituisce un puntatore al pulsante creato, che può essere utilizzato per aggiungere eventi **LV\_EVENT\_CLICKED** (o altri) per rilevare e agire sulla risposta dell'utente.

I pulsanti del piè di pagina così aggiunti sono equidistanti e centrati.

È anche possibile aggiungere pulsanti all'intestazione con *lv\_msgbox\_add\_header\_button (msgbox, symbol)*. I pulsanti aggiunti in questo modo vengono aggiunti all'estremità destra dell'intestazione.

## Ottenere le parti

I componenti costitutivi della finestra di messaggio possono essere ottenuti utilizzando le seguenti funzioni:

```
lv_obj_t * lv_msgbox_get_content(lv_obj_t * msgbox);
lv_obj_t * lv_msgbox_get_title(lv_obj_t * msgbox);
lv_obj_t * lv_msgbox_get_header(lv_obj_t * msgbox);
lv_obj_t * lv_msgbox_get_footer(lv_obj_t * msgbox);
```

Le funzioni che aggiungono qualcosa alla finestra di messaggio restituiscono un puntatore al Widget appena aggiunto:

```
lv_obj_t * lv_msgbox_add_text(lv_obj_t * msgbox, const char * text);
lv_obj_t * lv_msgbox_add_title(lv_obj_t * msgbox, const char * title);
lv_obj_t * lv_msgbox_add_close_button(lv_obj_t * msgbox);
lv_obj_t * lv_msgbox_add_header_button(lv_obj_t * msgbox, const void * symbol);
lv_obj_t * lv_msgbox_add_footer_button(lv_obj_t * msgbox, const char * text);
```

## Chiudere la finestra di messaggio

`lv_msgbox_close(msgbox)` chiude (elimina) la finestra di messaggio.

`lv_msgbox_close_async(msgbox)` chiude (elimina) la finestra di messaggio in modo asincrono. Questo è utile se si desidera che la finestra di messaggio si chiuda alla successiva chiamata a `lv_timer_handler` anziché immediatamente.

### 6.23.4 Eventi

I Widget "Message Box" non inviano eventi speciali. Per i dettagli, consultare la documentazione di questi widget: *Button* (`lv_button`) e *Label* (`lv_label`).

#### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

### 6.23.5 Tasti

Nessun *Tasto* viene elaborato dai Widget "Message Box".

#### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

### 6.23.6 Esempio

#### Casella messaggi semplice

```
#include "../../lv_examples.h"
#if LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * label = lv_obj_get_child(btn, 0);
    LV_UNUSED(label);
    LV_LOG_USER("Button %s clicked", lv_label_get_text(label));
}

void lv_example_msgbox_1(void)
{
    lv_obj_t * mbox1 = lv_msgbox_create(NULL);

    lv_msgbox_add_title(mbox1, "Hello");

    lv_msgbox_add_text(mbox1, "This is a message box with two buttons.");
    lv_msgbox_add_close_button(mbox1);

    lv_obj_t * btn;
    btn = lv_msgbox_add_footer_button(mbox1, "Apply");
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);
    btn = lv_msgbox_add_footer_button(mbox1, "Cancel");
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);
    return;
}
```

(continues on next page)

(continua dalla pagina precedente)

#endif

## Casella messaggi con scorrimento e stile

```
#include "../../lv_examples.h"
#if LV_USE_MSGBOX && LV_BUILD_EXAMPLES

static void minimize_button_event_cb(lv_event_t * e)
{
    lv_obj_t * mbox = (lv_obj_t *) lv_event_get_user_data(e);
    lv_obj_add_flag(mbox, LV_OBJ_FLAG_HIDDEN);
}

void lv_example_msgbox_2(void)
{
    lv_obj_t * setting = lv_msgbox_create(lv_screen_active());
    lv_obj_set_style_clip_corner(setting, true, 0);

    /* setting fixed size */
    lv_obj_set_size(setting, 300, 200);

    /* setting's titlebar/header */
    lv_msgbox_add_title(setting, "Setting");
    lv_obj_t * minimize_button = lv_msgbox_add_header_button(setting, LV_SYMBOL_MINUS);
    lv_obj_add_event_cb(minimize_button, minimize_button_event_cb, LV_EVENT_CLICKED, setting);
    lv_msgbox_add_close_button(setting);

    /* setting's content*/
    lv_obj_t * content = lv_msgbox_get_content(setting);
    lv_obj_set_flex_flow(content, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(content, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj_set_style_pad_right(content, -1, LV_PART_SCROLLBAR);

    lv_obj_t * cont_brightness = lv_obj_create(content);
    lv_obj_set_size(cont_brightness, lv_pct(100), LV_SIZE_CONTENT);
    lv_obj_set_flex_flow(cont_brightness, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(cont_brightness, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * lb_brightness = lv_label_create(cont_brightness);
    lv_label_set_text(lb_brightness, "Brightness : ");
    lv_obj_t * slider_brightness = lv_slider_create(cont_brightness);
    lv_obj_set_width(slider_brightness, lv_pct(100));
    lv_slider_set_value(slider_brightness, 50, LV_ANIM_OFF);

    lv_obj_t * cont_speed = lv_obj_create(content);
    lv_obj_set_size(cont_speed, lv_pct(100), LV_SIZE_CONTENT);
    lv_obj_set_flex_flow(cont_speed, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(cont_speed, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * lb_speed = lv_label_create(cont_speed);
    lv_label_set_text(lb_speed, "Speed : ");
    lv_obj_t * slider_speed = lv_slider_create(cont_speed);
    lv_obj_set_width(slider_speed, lv_pct(100));
    lv_slider_set_value(slider_speed, 80, LV_ANIM_OFF);

    /* footer */
    lv_obj_t * apply_button = lv_msgbox_add_footer_button(setting, "Apply");
    lv_obj_set_flex_grow(apply_button, 1);

    lv_obj_t * cancel_button = lv_msgbox_add_footer_button(setting, "Cancel");
    lv_obj_set_flex_grow(cancel_button, 1);

    lv_obj_t * footer = lv_msgbox_get_footer(setting);
    lv_obj_set_style_bg_color(footer, lv_palette_main(LV_PALETTE_INDIGO), 0);
    lv_obj_set_style_bg_opa(footer, LV_OPA_100, 0);
}
#endif
```

## 6.23.7 API

*lv\_msgbox.h*

*lv\_msgbox\_private.h*

## 6.24 Roller (lv\_roller)

### 6.24.1 Panoramica

Il Roller consente all'utente finale di selezionare un elemento da una lista scorrendola. L'elemento al centro è quello selezionato e normalmente si distingue dagli altri per i diversi stili applicati.

### 6.24.2 Parti e Stili

- **LV\_PART\_MAIN** Il background del Roller utilizza le proprietà *tipiche del background* e lo stile del testo.
  - Lo stile **text\_line\_space** regola lo spazio tra gli elementi della lista. Si usa **lv\_obj\_set\_style\_text\_line\_space()** per impostare questo valore.

- Quando il Roller scorre e non si ferma esattamente su un elemento, scorrerà automaticamente fino all'elemento valido più vicino in `anim_time` millisecondi, come specificato nello stile `anim_duration`. Si usa `lv_obj_set_style_anim_duration()` per impostare questo valore.
- `LV_PART_SELECTED` L'elemento selezionato (visualizzato al centro del Roller). Oltre alle tipiche proprietà di background, utilizza le proprietà di stile del testo per modificare l'aspetto del testo dell'elemento selezionato.

### 6.24.3 Utilizzo

#### Impostazione degli elementi della lista

Gli elementi della lista vengono passati al Roller come stringa con `lv_roller_set_options(roller, string_list, LV_ROLLER_MODE_NORMAL)`. Gli elementi devono essere separati da \n. Per esempio: "First\nSecond\nThird".

`LV_ROLLER_MODE_INFINITE` rende il Roller circolare.

È possibile selezionare un'opzione a livello di codice con `lv_roller_set_selected(roller, id, LV_ANIM_ON)`, dove `id` è l'indice a partire da zero dell'elemento della lista da selezionare.

Se non si conosce l'indice di un'opzione, è possibile selezionare un elemento anche con `lv_roller_set_selected_str(roller, str, LV_ANIM_ON)`, dove `str` la stringa corrispondente a uno degli elementi dell'elenco.

#### Prelevare l'opzione selezionata

Per ottenere l'*indice* dell'elemento attualmente selezionato, si usa `lv_roller_get_selected(roller)`.

`lv_roller_get_selected_str(roller, buf, buf_size)` copierà il nome dell'elemento selezionato in `buf`.

#### Righe visibili

Il numero di righe visibili può essere regolato con `lv_roller_set_visible_row_count(roller, num)`.

Questa funzione calcola l'altezza con lo stile corrente. Se il font, l'interlinea, la larghezza del bordo, ecc. del Roller cambiano, questa funzione deve essere richiamata nuovamente.

### 6.24.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando viene selezionato un nuovo elemento dell'elenco.

#### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

### 6.24.5 Tasti

- `LV_KEY_RIGHT/DOWN` Seleziona l'opzione successiva
- `LV_KEY_LEFT/UP` Seleziona l'opzione precedente
- `LY_KEY_ENTER` Accetta l'opzione selezionata (invia l'evento `LV_EVENT_VALUE_CHANGED`)

#### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

## 6.24.6 Esempio

### Roller semplice

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected month: %s\n", buf);
    }
}

/**
 * An infinite roller with the name of the months
 */
void lv_example_roller_1(void)
{
    lv_obj_t * roller1 = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller1,
        "January\n"
        "February\n"
        "March\n"
        "April\n"
        "May\n"
        "June\n"
        "July\n"
        "August\n"
        "September\n"
        "October\n"
        "November\n"
        "December",
        LV_ROLLER_MODE_INFINITE);

    lv_roller_set_visible_row_count(roller1, 4);
    lv_obj_center(roller1);
    lv_obj_add_event_cb(roller1, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

### Roller con stili

```
#include "../../lv_examples.h"
#if LV_USE_ROLLER && LV_FONT_MONTserrat_22 && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_roller_get_selected_str(obj, buf, sizeof(buf));
        LV_LOG_USER("Selected value: %s", buf);
    }
}

/**
 * Roller with various alignments and larger text in the selected area
 */
void lv_example_roller_2(void)
{
    /*A style to make the selected option larger*/
    static lv_style_t style_sel;
    lv_style_init(&style_sel);
    lv_style_set_text_font(&style_sel, &lv_font_montserrat_22);
    lv_style_set_bg_color(&style_sel, lv_color_hex3(0xf88));
    lv_style_set_border_width(&style_sel, 2);
    lv_style_set_border_color(&style_sel, lv_color_hex3(0xf00));

    const char * opts = "1\n2\n3\n4\n5\n6\n7\n8\n9\n10";
    lv_obj_t * roller;

    /*A roller on the left with left aligned text, and custom width*/
    roller = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 2);
    lv_obj_set_width(roller, 100);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_LEFT, 0);
    lv_obj_set_style_bg_color(roller, lv_color_hex3(0x0f0), 0);
    lv_obj_set_style_bg_grad_color(roller, lv_color_hex3(0xaafa), 0);
    lv_obj_set_style_bg_grad_dir(roller, LV_GRAD_DIR_VER, 0);
    lv_obj_align(roller, LV_ALIGN_LEFT_MID, 10, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 2, LV_ANIM_OFF);

    /*A roller on the middle with center aligned text, and auto (default) width*/
    roller = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 3);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_align(roller, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
    lv_roller_set_selected(roller, 5, LV_ANIM_OFF);

    /*A roller on the right with right aligned text, and custom width*/
    roller = lv_roller_create(lv_screen_active());
    lv_roller_set_options(roller, opts, LV_ROLLER_MODE_NORMAL);
    lv_roller_set_visible_row_count(roller, 4);
    lv_obj_set_width(roller, 80);
    lv_obj_add_style(roller, &style_sel, LV_PART_SELECTED);
    lv_obj_set_style_text_align(roller, LV_TEXT_ALIGN_RIGHT, 0);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_align(roller, LV_ALIGN_RIGHT_MID, -10, 0);
lv_obj_add_event_cb(roller, event_handler, LV_EVENT_ALL, NULL);
lv_roller_set_selected(roller, 8, LV_ANIM_OFF);
}

#endif

```

## Aggiungere una maschera di dissolvenza al roller

```

#include "../../lv_examples.h"

#if LV_USE_ROLLER && LV_DRAW_SW_COMPLEX && LV_BUILD_EXAMPLES && 1

static void generate_mask(lv_draw_buf_t * mask)
{
    /*Create a "8 bit alpha" canvas and clear it*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, mask);
    lv_canvas_fill_bg(canvas, lv_color_white(), LV_OPA_TRANSP);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    /*Draw a label to the canvas. The result "image" will be used as mask*/
    lv_draw_rect_dsc_t rect_dsc;
    lv_draw_rect_dsc_init(&rect_dsc);
    rect_dsc.bg_grad.dir = LV_GRAD_DIR_VER;
    rect_dsc.bg_grad.stops[0].color = lv_color_black();
    rect_dsc.bg_grad.stops[1].color = lv_color_white();
    rect_dsc.bg_grad.stops[0].opa = LV_OPA_COVER;
    rect_dsc.bg_grad.stops[1].opa = LV_OPA_COVER;
    lv_area_t a = {0, 0, mask->header.w - 1, mask->header.h / 2 - 10};
    lv_draw_rect(&layer, &rect_dsc, &a);

    a.y1 = mask->header.h / 2 + 10;
    a.y2 = mask->header.h - 1;
    rect_dsc.bg_grad.stops[0].color = lv_color_white();
    rect_dsc.bg_grad.stops[1].color = lv_color_black();
    lv_draw_rect(&layer, &rect_dsc, &a);

    lv_canvas_finish_layer(canvas, &layer);

    /*Comment it to make the mask visible*/
    lv_obj_delete(canvas);
}

/**
 * Add a fade mask to roller.
 */
void lv_example_roller_3(void)
{
    lv_obj_set_style_bg_color(lv_screen_active(), lv_palette_main(LV_PALETTE_BLUE_GREY), 0);

    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_bg_color(&style, lv_color_black());
    lv_style_set_text_color(&style, lv_color_white());
    lv_style_set_border_width(&style, 0);
    lv_style_set_radius(&style, 0);

    lv_obj_t * roller1 = lv_roller_create(lv_screen_active());
    lv_obj_add_style(roller1, &style, 0);
    lv_obj_set_style_bg_opa(roller1, LV_OPA_50, LV_PART_SELECTED);

    lv_roller_set_options(roller1,
        "January\n"
        "February\n"
        "March\n"
        "April\n"
        "May\n"
        "June\n"
        "July\n"
        "August\n"
        "September\n"
        "October\n"
        "November\n"
        "December",
        LV_ROLLER_MODE_NORMAL);

    lv_obj_center(roller1);
    lv_roller_set_visible_row_count(roller1, 4);

    /* Create the mask to make the top and bottom part of roller faded.
     * The width and height are empirical values for simplicity*/
    LV_DRAW_BUF_DEFINE_STATIC(mask, 130, 150, LV_COLOR_FORMAT_L8);
    LV_DRAW_BUF_INIT_STATIC(mask);

    generate_mask(&mask);
    lv_obj_set_style_bitmap_mask_src(roller1, &mask, 0);
}
#endif

```

## 6.24.7 API

*lv\_api\_map\_v8.h*

*lv\_observer.h*

*lv\_obj\_property\_names.h*

*lv\_roller.h*

*lv\_roller\_private.h*

## 6.25 Scale (lv\_scale)

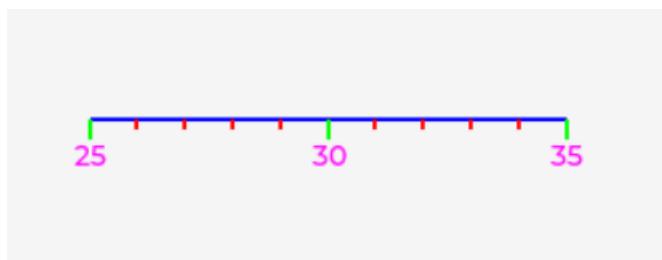
### 6.25.1 Panoramica

I Widget "Scala" mostrano scale lineari o circolari con intervalli, conteggi di tacche, posizionamento, etichettatura e sottosezioni (*Sezioni*) configurabili con stile personalizzato.

### 6.25.2 Parti e Stili

Il Widget Scala è composto dalle seguenti tre parti:

- *LV\_PART\_MAIN* Linea principale --- la linea blu nell'immagine di esempio.
- *LV\_PART\_ITEMS* Tacche secondarie --- le tacche secondarie rosse nell'immagine di esempio.
- *LV\_PART\_INDICATOR* Tacche principali e relative etichette (se abilitate) --- le tacche principali verdi e le etichette rosa nell'immagine di esempio.



### 6.25.3 Utilizzo

#### Modalità

Quando viene creato un Widget Scala, inizia in MODALITÀ *LV\_SCALE\_MODE\_HORIZONTAL\_BOTTOM*. Questo rende la scala orizzontale con le tacche sotto la linea. Se c'è bisogno di una forma, un orientamento o una posizione di tacca diversa, si usa *lv\_scale\_set\_mode(scale, mode)*, dove *mode* può essere uno qualsiasi di questi valori:

- *LV\_SCALE\_MODE\_HORIZONTAL\_TOP*
- *LV\_SCALE\_MODE\_HORIZONTAL\_BOTTOM*
- *LV\_SCALE\_MODE\_VERTICAL\_LEFT*
- *LV\_SCALE\_MODE\_VERTICAL\_RIGHT*
- *LV\_SCALE\_MODE\_ROUND\_INNER*
- *LV\_SCALE\_MODE\_ROUND\_OUTER*

#### Impostazione del range

Una Scala inizia la sua vita con un intervallo numerico predefinito di [0..100] e un intervallo angolare predefinito di 270. È possibile modificare questi intervalli con:

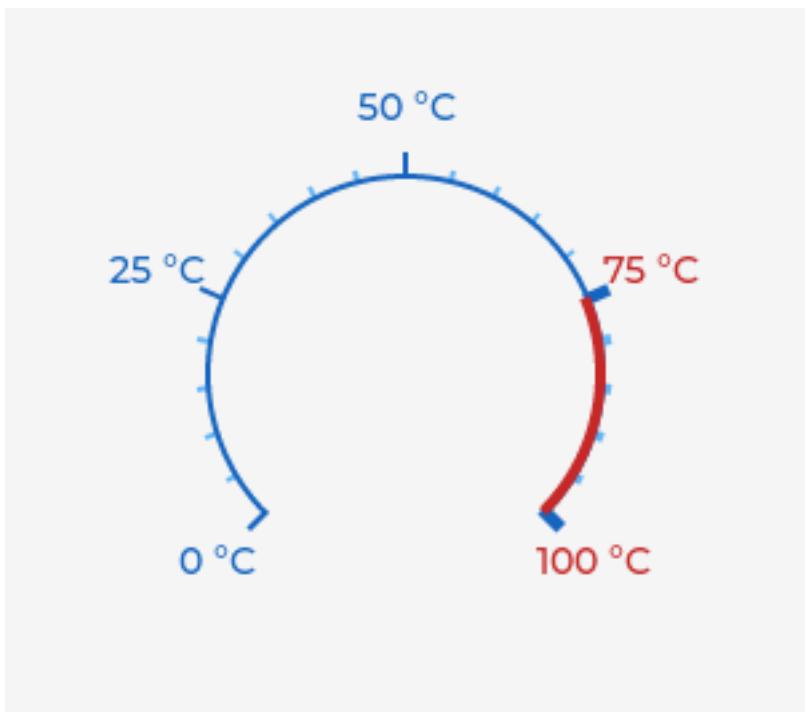
- *lv\_scale\_set\_range(scale, min, max)*, e
- *lv\_scale\_set\_angle\_range(scale, angle\_range)*

dove *min* e *max* diventeranno i valori numerici minimo e massimo della Scale, e *angle\_range* è l'angolo tra i limiti minimo e massimo della Scale.

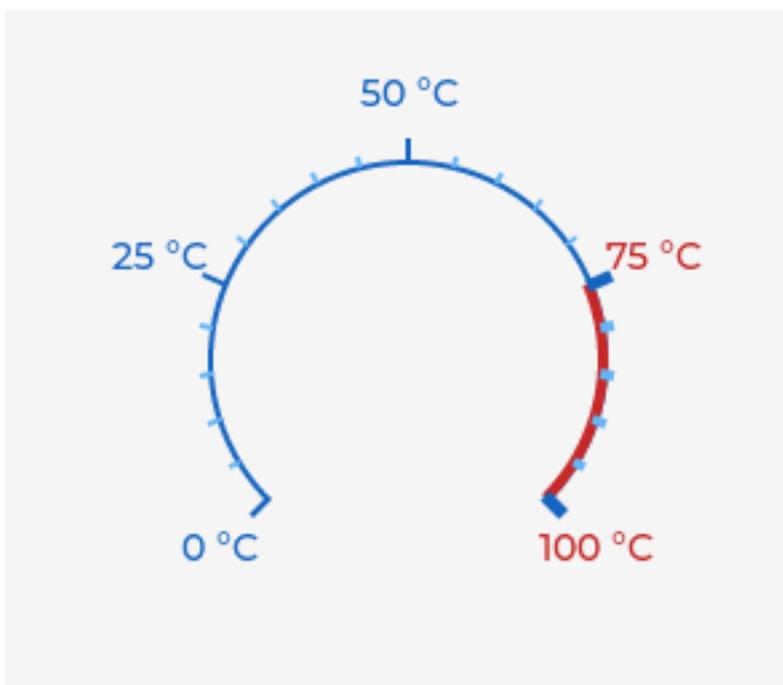
### Ordine di disegno delle tacche

Normalmente, le tacche e le relative etichette vengono disegnate per prime, seguite dalla linea principale, dando l'impressione che le tacche e le relative etichette siano posizionate sotto la linea principale quando si sovrappongono. È possibile invertire questa sequenza, se lo si desidera, facendo apparire le tacche e le etichette sopra la linea principale, utilizzando `lv_scale_set_draw_ticks_on_top(scale, true)`. (Questo effetto può essere invertito passando `false`.)

Esempio con tacche ed etichette disegnate *sotto* la linea principale (il default):



Esempio con tacche ed etichette disegnate *sopra* la linea principale:



## Configurazione delle tacche

È possibile configurare le tacche maggiori e minori di una Scala chiamando 2 funzioni:

- `lv_scale_set_total_tick_count(scale, total_tick_count)` e
- `lv_scale_set_major_tick_every(scale, nth_tick)`.

Per far sì che le etichette vengano disegnate con le tacche maggiori, si chiama `lv_scale_set_label_show(scale, true)`. (Passare `false` per nasconderle di nuovo).

Per default, le etichette mostrate sono i valori numerici della scala nei punti delle tacche maggiori. È possibile specificare un contenuto di etichetta diverso chiamando `lv_scale_set_text_src(scale, custom_labels)` dove `custom_labels` è un array di puntatori a stringa. Esempio:

```
static char * custom_labels[3] = {"One", "Two", NULL};
```

Si noti che per terminare l'elenco viene utilizzato un puntatore `NULL`.

Il contenuto dei buffer a cui si punta deve rimanere valido per tutta la durata della Scala.

Per una Scala in una delle modalità `..._ROUND_...`, è possibile facoltativamente far ruotare le etichette dei tick principali in modo che corrispondano alla rotazione degli stessi utilizzando `lv_obj_set_style_transform_rotation(scale, LV_SCALE_LABEL_ROTATE_MATCH_TICKS, LV_PART_INDICATOR)`.

In alternativa, le etichette possono essere ruotate di un valore fisso (per qualsiasi modalità della Scala). Questo esempio ruota le etichette di 20 gradi: `lv_obj_set_style_transform_rotation(scale, 200, LV_PART_INDICATOR)`.

Oppure è possibile eseguire entrambe le operazioni contemporaneamente: `lv_obj_set_style_transform_rotation(scale, LV_SCALE_LABEL_ROTATE_MATCH_TICKS + 200, LV_PART_INDICATOR)`.

Alcune etichette della Scala potrebbero essere disegnate capovolte (per corrispondere alla tacca) se la Scala include un certo intervallo di angoli. Se non si desidera che ciò avvenga, è possibile utilizzare un flag aggiuntivo per ruotare automaticamente le etichette e mantenerle in posizione verticale. Le etichette che sarebbero capovolte vengono quindi ruotate di 180 `lv_obj_set_style_transform_rotation(scale, LV_SCALE_LABEL_ROTATE_MATCH_TICKS | LV_SCALE_LABEL_ROTATE_KEEP_UPRIGHT, LV_PART_INDICATOR)`. Le etichette possono anche essere spostate di una distanza fissa in pixel X e Y utilizzando `lv_obj_set_style_translate_x(scale, 10, LV_PART_INDICATOR)`.

### Nota

Il valore della tacca principale viene calcolato con l'API `lv_map` (quando non si impostano etichette personalizzate); questo calcolo tiene conto del numero totale di tacche e dell'intervallo della Scala, quindi l'etichetta disegnata può presentare errori di arrotondamento quando il valore calcolato è un valore in virgola mobile.

La lunghezza delle tacche può essere configurata con la proprietà "length Style" su `LV_PART_INDICATOR` per le tacche principali e `LV_PART_ITEMS` per le tacche secondarie. Esempio con Stile locale: `lv_obj_set_style_length(scale, 5, LV_PART_INDICATOR)` per le tacche principali e `lv_obj_set_style_length(scale, 5, LV_PART_ITEMS)` per le tacche secondarie. Le tacche possono essere riempite in entrambe le direzioni (verso l'esterno o verso l'interno) per le Scale `..._ROUND_...` solo con: `lv_obj_set_style_radial_offset(scale, 5, LV_PART_INDICATOR)` per le tacche principali e `lv_obj_set_style_radial_offset(scale, 5, LV_PART_ITEMS)` per quelle secondarie. L'utilizzo combinato di lunghezza e offset radiale consente il controllo totale della posizione della tacca.

È anche possibile compensare le etichette dai segni principali (positivi o negativi) utilizzando `lv_obj_set_style_pad_radial(scale, 5, LV_PART_INDICATOR)`

## Sezioni

Le Sezioni consentono a porzioni di una Scala di *trasmettere significato* utilizzando diverse proprietà di Stile per disegnarle (colori, spessori delle linee, font, ecc.).

Una Sezione rappresenta un sottointervallo della Scala, i cui Stili (come i Cascading Style Sheets) hanno la precedenza durante il disegno delle PARTI (linee, archi, tacche ed etichette) della Scala che rientrano nell'intervallo di quella Sezione.

Se una PARTE di una Scala rientra nell'intervallo di 2 o più Sezioni (ovvero se tali Sezioni si sovrappongono), le proprietà dello Stile appartenenti alla Sezione aggiunta più di recente hanno la precedenza sulle stesse proprietà di stile delle altre Sezioni che "coinvolgono" quella PARTE.

### Creazione di Sezioni

Una sezione viene creata utilizzando `lv_scale_add_section(scal`e), che restituisce un puntatore a un oggetto `lv_scale_section_t` object. Questo crea una Sezione con intervallo [0.0] e senza Stili aggiuntivi, il che garantisce che la Sezione non venga ancora disegnata: necessita sia di un intervallo all'interno dell'intervallo della Scala che di almeno uno *Stile* aggiunto prima di poter essere utilizzata nel disegno della Scala.

Successivamente, si imposta l'intervallo usando `lv_scale_section_set_range(section, min, max)` dove `min` e `max` sono i valori limite della Sezione che normalmente dovrebbero essere compresi nell'intervallo di valori della Scala. (Se rientrano solo parzialmente nell'intervallo della Scala, la Scala utilizzerà solo la porzione della Sezione che si sovrappone all'intervallo della Scala. Se l'intervallo di una Sezione non rientra affatto nell'intervallo della Scala, non verrà utilizzato nel disegno. Questo può essere utile per "disattivare" temporaneamente una Sezione, ad esempio `lv_scale_section_set_range(section, 0, -1)`.)

### Stili delle Sezioni

Per impostare le proprietà di Stile di una Sezione, è possibile creare un oggetto `lv_style_t` per ogni "sezione" che si desidera visualizzare in modo diverso rispetto alla Scala padre. Aggiungere le proprietà di stile come documentato in *Inizializzare gli Stili e Impostare/Ottenere Proprietà*.

Aggiungere ciascun oggetto `lv_style_t` a ogni Sezione a cui verrà applicato utilizzando `lv_scale_section_set_style(section, PART, style_pointer)`, dove:

- `style_pointer` dovrebbe puntare al contenuto di una variabile globale o statica (può essere allocata dinamicamente), poiché deve rimanere valida per tutta la durata della Scala; e
- `PART` indica a quale singola *PART* della Scala padre verrà applicato, ovvero `LV_PART_MAIN`, `LV_PART_ITEMS` o `LV_PART_INDICATOR`.

A differenza dell'aggiunta di stili normali ai Widget, non è possibile combinare le PARTI eseguendo l'operazione OR bit per bit sui valori delle PARTI per ottenere che lo stile venga applicato a più di una parte. Tuttavia, è possibile fare qualcosa di simile per ottenere lo stesso risultato:

```
static lv_style_t tick_style;
lv_style_init(&tick_style);
lv_style_set_line_color(&tick_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_scale_section_set_style(section, LV_PART_ITEMS, &tick_style);
lv_scale_section_set_style(section, LV_PART_INDICATOR, &tick_style);
```

per fare in modo che quell'oggetto Style venga applicato sia ai tick maggiori che a quelli minori.

Gli oggetti `lv_style_t` possono essere condivisi tra le Sezioni e tra le PARTI, ma a differenza dei normali Stili aggiuntivi a un Widget, una Sezione può avere solo 1 stile per PARTE. Quindi, procedendo in questo modo:

```
lv_scale_section_set_style(section, LV_PART_INDICATOR, &tick_style_1);
lv_scale_section_set_style(section, LV_PART_INDICATOR, &tick_style_2);
```

sostituisce `tick_style_1` con `tick_style_2` per la parte `LV_PART_INDICATOR` anziché aggiungerla.

## Proprietà di Stile utili per le Sezioni

Le proprietà di Stile utilizzate durante il disegno della Scala (e quindi utili) sono:

- Per la linea principale *quando è una linea retta* ([LV\\_PART\\_MAIN](#)):

**LV\_STYLE\_LINE\_WIDTH**  
`lv_style_set_line_width()`

**LV\_STYLE\_LINE\_COLOR**  
`lv_style_set_line_color()`

**LV\_STYLE\_LINE\_OPA**  
`lv_style_set_line_opa()`

- Per la linea principale *quando è un arco* ([LV\\_PART\\_MAIN](#)):

**LV\_STYLE\_ARC\_WIDTH**  
`lv_style_set_arc_width()`

**LV\_STYLE\_ARC\_COLOR**  
`lv_style_set_arc_color()`

**LV\_STYLE\_ARC\_OPA**  
`lv_style_set_arc_opa()`

**LV\_STYLE\_ARC\_ROUNDED**  
`lv_style_set_arc_rounded()`

**LV\_STYLE\_ARC\_IMAGE\_SRC**  
`lv_style_set_arc_image_src()`

- Per le linee delle tacche ([LV\\_PART\\_ITEMS](#) e [LV\\_PART\\_INDICATOR](#)):

**LV\_STYLE\_LINE\_WIDTH**  
`lv_style_set_line_width()`

**LV\_STYLE\_LINE\_COLOR**  
`lv_style_set_line_color()`

**LV\_STYLE\_LINE\_OPA**  
`lv_style_set_line_opa()`

- Per le etichette sulle tacche principali ([LV\\_PART\\_INDICATOR](#))

**LV\_STYLE\_TEXT\_COLOR**  
`lv_style_set_text_color()`

**LV\_STYLE\_TEXT\_OPA**  
`lv_style_set_text_opa()`

**LV\_STYLE\_TEXT\_LETTER\_SPACE**  
`lv_style_set_text_letter_space()`

**LV\_STYLE\_TEXT\_FONT**  
`lv_style_set_text_font()`

## 6.25.4 Eventi

I Widget Scala non inviano eventi speciali.

Negli eventi [LV\\_EVENT\\_DRAW\\_TASK\\_ADDED](#), i membri `id1` e `id2` di un descrittore di disegno di linea principale o secondario saranno rispettivamente l'indice della tacca e il suo valore. Se la parte è [LV\\_PART\\_INDICATOR](#), si tratta di una tacca principale. Se la parte è [LV\\_PART\\_ITEMS](#), si tratta di una tacca secondaria.

### **i** Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

## 6.25.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Scala.

### **i** Ulteriori Informazioni

Scoprire di più sui *Tasti*.

## 6.25.6 Esempio

### Una semplice scala orizzontale

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/**
 * A simple horizontal scale
 */
void lv_example_scale_1(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, lv_pct(80), 100);
    lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 31);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 10, 40);
}

#endif
```

### Una scala verticale con sezioni e stile personalizzato

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/**
 * An vertical scale with section and custom styling
 */
void lv_example_scale_2(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 60, 200);
    lv_scale_set_label_show(scale, true);
    lv_scale_set_mode(scale, LV_SCALE_MODE_VERTICAL_RIGHT);
    lv_obj_center(scale);

    lv_scale_set_total_tick_count(scale, 21);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_scale_set_range(scale, 0, 100);

    static const char * custom_labels[] = {"0 °C", "25 °C", "50 °C", "75 °C", "100 °C", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);

    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));

    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
    lv_style_set_width(&indicator_style, 10U); /*Tick length*/
    lv_style_set_line_width(&indicator_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_palette_lighten(LV_PALETTE_BLUE, 2));
    lv_style_set_width(&minor_ticks_style, 5U); /*Tick length*/
    lv_style_set_line_width(&minor_ticks_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

static lv_style_t main_line_style;
lv_style_init(&main_line_style);
/* Main line properties */
lv_style_set_line_color(&main_line_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
lv_style_set_line_width(&main_line_style, 2U); /* Tick width
lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

/* Add a section */
static lv_style_t section_minor_tick_style;
static lv_style_t section_label_style;
static lv_style_t section_main_line_style;

lv_style_init(&section_label_style);
lv_style_init(&section_minor_tick_style);
lv_style_init(&section_main_line_style);

/* Label style properties */
lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
lv_style_set_text_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_line_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_line_width(&section_label_style, 5U); /*Tick width*/

lv_style_set_line_color(&section_minor_tick_style, lv_palette_lighten(LV_PALETTE_RED, 2));
lv_style_set_line_width(&section_minor_tick_style, 4U); /*Tick width*/

/* Main line properties */
lv_style_set_line_color(&section_main_line_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_line_width(&section_main_line_style, 4U); /*Tick width*/

/* Configure section styles */
lv_scale_section_t * section = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section, 75, 100);
lv_scale_set_section_style_indicator(scale, section, &section_label_style);
lv_scale_set_section_style_items(scale, section, &section_minor_tick_style);
lv_scale_set_section_style_main(scale, section, &section_main_line_style);

lv_obj_set_style_bg_color(scale, lv_palette_main(LV_PALETTE_BLUE_GREY), 0);
lv_obj_set_style_bg_opa(scale, LV_OPA_50, 0);
lv_obj_set_style_pad_left(scale, 8, 0);
lv_obj_set_style_radius(scale, 8, 0);
lv_obj_set_style_pad_ver(scale, 20, 0);
}

#endif

```

## Una semplice scala circolare

```

#include "../../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

LV_IMAGE_DECLARE(img_hand);

static lv_obj_t * needle_line;
static lv_obj_t * needle_img;

static void set_needle_line_value(void * obj, int32_t v)
{
    lv_scale_set_line_needle_value((lv_obj_t *)obj, needle_line, 60, v);
}

static void set_needle_img_value(void * obj, int32_t v)
{
    lv_scale_set_image_needle_value((lv_obj_t *)obj, needle_img, v);
}

/**
 * A simple round scale
 */
void lv_example_scale_3(void)
{
    lv_obj_t * scale_line = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale_line, 150, 150);
    lv_scale_set_mode(scale_line, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale_line, LV_OPA_COVER, 0);
    lv_obj_set_style_bg_color(scale_line, lv_palette_lighten(LV_PALETTE_RED, 5), 0);
    lv_obj_set_style_radius(scale_line, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(scale_line, true, 0);
    lv_obj_align(scale_line, LV_ALIGN_LEFT_MID, LV_PCT(2), 0);

    lv_scale_set_label_show(scale_line, true);

    lv_scale_set_total_tick_count(scale_line, 31);
    lv_scale_set_major_tick_every(scale_line, 5);

    lv_obj_set_style_length(scale_line, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale_line, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale_line, 10, 40);

    lv_scale_set_angle_range(scale_line, 270);
    lv_scale_set_rotation(scale_line, 135);

    needle_line = lv_line_create(scale_line);

    lv_obj_set_style_line_width(needle_line, 6, LV_PART_MAIN);
    lv_obj_set_style_line_rounded(needle_line, true, LV_PART_MAIN);

    lv_anim_t anim_scale_line;
    lv_anim_init(&anim_scale_line);
    lv_anim_set_var(&anim_scale_line, scale_line);
    lv_anim_set_exec_cb(&anim_scale_line, set_needle_line_value);
    lv_anim_set_duration(&anim_scale_line, 1000);
    lv_anim_set_repeat_count(&anim_scale_line, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_reverse_duration(&anim_scale_line, 1000);
    lv_anim_set_values(&anim_scale_line, 10, 40);
    lv_anim_start(&anim_scale_line);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * scale_img = lv_scale_create(lv_screen_active());
lv_obj_set_size(scale_img, 150, 150);
lv_scale_set_mode(scale_img, LV_SCALE_MODE_ROUND_INNER);
lv_obj_set_style_bg_opa(scale_img, LV_OPA_COVER, 0);
lv_obj_set_style_bg_color(scale_img, lv_palette_lighten(LV_PALETTE_RED, 5), 0);
lv_obj_set_style_radius(scale_img, LV_RADIUS_CIRCLE, 0);
lv_obj_set_style_clip_corner(scale_img, true, 0);
lv_obj_align(scale_img, LV_ALIGN_RIGHT_MID, LV_PCT(-2), 0);

lv_scale_set_label_show(scale_img, true);
lv_scale_set_total_tick_count(scale_img, 31);
lv_scale_set_major_tick_every(scale_img, 5);

lv_obj_set_style_length(scale_img, 5, LV_PART_ITEMS);
lv_obj_set_style_length(scale_img, 10, LV_PART_INDICATOR);
lv_scale_set_range(scale_img, 10, 40);

lv_scale_set_angle_range(scale_img, 270);
lv_scale_set_rotation(scale_img, 135);

/* image must point to the right. E.g. -0----->*
needle_img = lv_image_create(scale_img);
lv_image_set_src(needle_img, &img_hand);
lv_obj_align(needle_img, LV_ALIGN_CENTER, 47, -2);
lv_image_set_pivot(needle_img, 3, 4);

lv_anim_t anim_scale_img;
lv_anim_init(&anim_scale_img);
lv_anim_set_var(&anim_scale_img, scale_img);
lv_anim_set_exec_cb(&anim_scale_img, set_needle_img_value);
lv_anim_set_duration(&anim_scale_img, 1000);
lv_anim_set_repeat_count(&anim_scale_img, LV_ANIM_REPEAT_INFINITE);
lv_anim_set_reverse_duration(&anim_scale_img, 1000);
lv_anim_set_values(&anim_scale_img, 10, 40);
lv_anim_start(&anim_scale_img);
}

#endif

```

## Una scala circolare con sezione e stile personalizzato

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A round scale with section and custom styling
 */
void lv_example_scale_4(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 150, 150);
    lv_scale_set_label_show(scale, true);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_OUTER);
    lv_obj_center(scale);

    lv_scale_set_total_tick_count(scale, 21);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 0, 100);

    static const char * custom_labels[] = {"0 °C", "25 °C", "50 °C", "75 °C", "100 °C", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);

    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));

    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
    lv_style_set_width(&indicator_style, 10U); /*Tick length*/
    lv_style_set_line_width(&indicator_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_palette_lighten(LV_PALETTE_BLUE, 2));
    lv_style_set_width(&minor_ticks_style, 5U); /*Tick length*/
    lv_style_set_line_width(&minor_ticks_style, 2U); /*Tick width*/
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

    static lv_style_t main_line_style;
    lv_style_init(&main_line_style);
    /* Main line properties */
    lv_style_set_arc_color(&main_line_style, lv_palette_darken(LV_PALETTE_BLUE, 3));
    lv_style_set_arc_width(&main_line_style, 20); /*Tick width*/
    lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

    /* Add a section */
    static lv_style_t section_minor_tick_style;
    static lv_style_t section_label_style;
    static lv_style_t section_main_line_style;

    lv_style_init(&section_label_style);
    lv_style_init(&section_minor_tick_style);
    lv_style_init(&section_main_line_style);

    /* Label style properties */
    lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_line_color(&section_label_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_line_width(&section_label_style, 5U); /*Tick width*/

lv_style_set_line_color(&section_minor_tick_style, lv_palette_lighten(LV_PALETTE_RED, 2));
lv_style_set_line_width(&section_minor_tick_style, 4U); /*Tick width*/

/* Main line properties */
lv_style_set_arc_color(&section_main_line_style, lv_palette_darken(LV_PALETTE_RED, 3));
lv_style_set_arc_width(&section_main_line_style, 4U); /*Tick width*/

/* Configure section styles */
lv_scale_section_t * section = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section, 75, 100);
lv_scale_set_section_style_indicator(scale, section, &section_label_style);
lv_scale_set_section_style_items(scale, section, &section_minor_tick_style);
lv_scale_set_section_style_main(scale, section, &section_main_line_style);

}

#endif

```

## Una scala con sezione e stile personalizzato

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * An scale with section and custom styling
 */
void lv_example_scale_5(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, lv_display_get_horizontal_resolution(NULL) / 2, lv_display_get_vertical_resolution(NULL) / 2);
    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 10);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 25, 35);

    static const char * custom_labels[3] = {"One", "Two", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);
    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_color_hex(0xff00ff));
    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_color_hex(0x00ff00));
    lv_style_set_width(&indicator_style, 10U); // Tick length
    lv_style_set_line_width(&indicator_style, 2U); // Tick width
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_color_hex(0xffff00));
    lv_style_set_width(&minor_ticks_style, 5U); // Tick length
    lv_style_set_line_width(&minor_ticks_style, 2U); // Tick width
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

    static lv_style_t main_line_style;
    lv_style_init(&main_line_style);
    /* Main line properties */
    lv_style_set_line_color(&main_line_style, lv_color_hex(0x0000ff));
    lv_style_set_line_width(&main_line_style, 2U); // Tick width
    lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

    lv_obj_center(scale);

    /* Add a section */
    static lv_style_t section_minor_tick_style;
    static lv_style_t section_label_style;

    lv_style_init(&section_label_style);
    lv_style_init(&section_minor_tick_style);

    /* Label style properties */
    lv_style_set_text_font(&section_label_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&section_label_style, lv_color_hex(0xffff00));
    lv_style_set_text_letter_space(&section_label_style, 10);
    lv_style_set_text_opa(&section_label_style, LV_OPA_50);

    lv_style_set_line_color(&section_label_style, lv_color_hex(0xffff00));
    // lv_style_set_width(&section_label_style, 20U); // Tick length
    lv_style_set_line_width(&section_label_style, 5U); // Tick width

    lv_style_set_line_color(&section_minor_tick_style, lv_color_hex(0x0000ff));
    // lv_style_set_width(&section_label_style, 20U); // Tick length
    lv_style_set_line_width(&section_minor_tick_style, 4U); // Tick width

    /* Configure section styles */
    lv_scale_section_t * section = lv_scale_add_section(scale);
    lv_scale_set_section_range(scale, section, 25, 30);
    lv_scale_set_section_style_indicator(scale, section, &section_label_style);
    lv_scale_set_section_style_items(scale, section, &section_minor_tick_style);
}

#endif

```

## Una scala circolare con più lancette, simile a un orologio

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

#if LV_USE_FLOAT
#define my_PRIprecise "f"
#else
#define my_PRIprecise LV_PRId32
#endif

static lv_obj_t * scale;
static lv_obj_t * minute_hand;
static lv_obj_t * hour_hand;
static lv_point_precise_t minute_hand_points[2];
static int32_t hour;
static int32_t minute;

static void timer_cb(lv_timer_t * timer)
{
    LV_UNUSED(timer);

    minute++;
    if(minute > 59) {
        minute = 0;
        hour++;
        if(hour > 11) {
            hour = 0;
        }
    }

    /**
     * the scale will store the needle line points in the existing
     * point array if one was set with `lv_line_set_points_mutable`.
     * Otherwise, it will allocate the needle line points.
     */

    /* the scale will store the minute hand line points in `minute_hand_points` */
    lv_scale_set_line_needle_value(scale, minute_hand, 60, minute);
    /* log the points that were stored in the array */
    LV_LOG_USER(
        "minute hand points - "
        "0: (%" PRIprecise ", %" PRIprecise "), "
        "1: (%" PRIprecise ", %" PRIprecise ")",
        minute_hand_points[0].x, minute_hand_points[0].y,
        minute_hand_points[1].x, minute_hand_points[1].y
    );

    /* the scale will allocate the hour hand line points */
    lv_scale_set_line_needle_value(scale, hour_hand, 40, hour * 5 + (minute / 12));
}

/**
 * A round scale with multiple needles, resembling a clock
 */
void lv_example_scale_6(void)
{
    scale = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale, 150, 150);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale, LV_OPA_60, 0);
    lv_obj_set_style_bg_color(scale, lv_color_black(), 0);
    lv_obj_set_style_radius(scale, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_clip_corner(scale, true, 0);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);

    lv_scale_set_total_tick_count(scale, 61);
    lv_scale_set_major_tick_every(scale, 5);

    static const char * hour_ticks[] = {"12", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", NULL};
    lv_scale_set_text_src(scale, hour_ticks);

    static lv_style_t indicator_style;
    lv_style_init(&indicator_style);

    /* Label style properties */
    lv_style_set_text_font(&indicator_style, LV_FONT_DEFAULT);
    lv_style_set_text_color(&indicator_style, lv_palette_main(LV_PALETTE_YELLOW));

    /* Major tick properties */
    lv_style_set_line_color(&indicator_style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_length(&indicator_style, 8); /* tick length */
    lv_style_set_line_width(&indicator_style, 2); /* tick width */
    lv_obj_add_style(scale, &indicator_style, LV_PART_INDICATOR);

    /* Minor tick properties */
    static lv_style_t minor_ticks_style;
    lv_style_init(&minor_ticks_style);
    lv_style_set_line_color(&minor_ticks_style, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_length(&minor_ticks_style, 6); /* tick length */
    lv_style_set_line_width(&minor_ticks_style, 2); /* tick width */
    lv_obj_add_style(scale, &minor_ticks_style, LV_PART_ITEMS);

    /* Main line properties */
    static lv_style_t main_line_style;
    lv_style_init(&main_line_style);
    lv_style_set_arc_color(&main_line_style, lv_color_black());
    lv_style_set_arc_width(&main_line_style, 5);
    lv_obj_add_style(scale, &main_line_style, LV_PART_MAIN);

    lv_scale_set_range(scale, 0, 60);

    lv_scale_set_angle_range(scale, 360);
    lv_scale_set_rotation(scale, 270);

    minute_hand = lv_line_create(scale);
    lv_line_set_points_mutable(minute_hand, minute_hand_points, 2);

    lv_obj_set_style_line_width(minute_hand, 3, 0);
    lv_obj_set_style_line_rounded(minute_hand, true, 0);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_style_line_color(minute_hand, lv_color_white(), 0);
hour_hand = lv_line_create(scale);

lv_obj_set_style_line_width(hour_hand, 5, 0);
lv_obj_set_style_line_rounded(hour_hand, true, 0);
lv_obj_set_style_line_color(hour_hand, lv_palette_main(LV_PALETTE_RED), 0);

hour = 11;
minute = 5;
lv_timer_t * timer = lv_timer_create(timer_cb, 250, NULL);
lv_timer_ready(timer);
}

#endif

```

## Personalizzazione del colore dell'etichetta principale della scala con l'evento LV\_EVENT\_DRAW\_TASK\_ADDED

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

#include "../../../lvgl_private.h" /*To expose the fields of lv_draw_task_t*/

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        if(label_draw_dsc) {
            const lv_color_t color_idx[7] = {
                lv_palette_main(LV_PALETTE_RED),
                lv_palette_main(LV_PALETTE_ORANGE),
                lv_palette_main(LV_PALETTE_YELLOW),
                lv_palette_main(LV_PALETTE_GREEN),
                lv_palette_main(LV_PALETTE_CYAN),
                lv_palette_main(LV_PALETTE_BLUE),
                lv_palette_main(LV_PALETTE_PURPLE),
            };
            uint32_t major_tick = lv_scale_get_major_tick_every(obj);
            label_draw_dsc->color = color_idx[base_dsc->id1 / major_tick];

            /*Free the previously allocated text if needed*/
            if(label_draw_dsc->text_local) lv_free((void *)label_draw_dsc->text);

            /*Malloc the text and set text_local as 1 to make LVGL automatically free the text.
             * (Local texts are malloc'd internally by LVGL. Mimic this behavior here too)*/
            char tmp_buffer[20] = {0}; /* Big enough buffer */
            lv_snprintf(tmp_buffer, sizeof(tmp_buffer), "%lf", (double)base_dsc->id2);
            label_draw_dsc->text = lv_strdup(tmp_buffer);
            label_draw_dsc->text_local = 1;

            lv_text_attributes_t attributes = {0};
            lv_point_t size;

            attributes.text_flags = LV_TEXT_FLAG_NONE;
            attributes.letter_space = 0;
            attributes.line_space = 0;
            attributes.max_width = 1000;

            lv_text_get_size(&size, label_draw_dsc->text, label_draw_dsc->font, &attributes);
            int32_t new_w = size.x;
            int32_t old_w = lv_area_get_width(&draw_task->area);

            /* Distribute the new size equally on both sides */
            draw_task->area.x1 -= (new_w - old_w) / 2;
            draw_task->area.x2 += ((new_w - old_w) + 1) / 2; /* +1 for rounding */
        }
    }

    /**
     * Customizing scale major tick label color with `LV_EVENT_DRAW_TASK_ADDED` event
     */
    void lv_example_scale_7(void)
    {
        lv_obj_t * scale = lv_scale_create(lv_screen_active());
        lv_obj_set_size(scale, lv_pct(80), 100);
        lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
        lv_obj_center(scale);

        lv_scale_set_label_show(scale, true);

        lv_scale_set_total_tick_count(scale, 31);
        lv_scale_set_major_tick_every(scale, 5);

        lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
        lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
        lv_scale_set_range(scale, 10, 40);

        lv_obj_add_event_cb(scale, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
        lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
    }
}

```

## Una scala circolare con etichette ruotate e traslate

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A simple round scale with label/tick translation
 */
void lv_example_scale_8(void)
{
    lv_obj_t * scale_line = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale_line, 150, 150);
    lv_scale_set_mode(scale_line, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_style_bg_opa(scale_line, LV_OPA_COVER, 0);
    lv_obj_set_style_bg_color(scale_line, lv_palette_lighten(LV_PALETTE_RED, 5), 0);
    lv_obj_set_style_radius(scale_line, LV_RADIUS_CIRCLE, 0);
    lv_obj_align(scale_line, LV_ALIGN_LEFT_MID, LV_PCT(2), 0);

    /*Set the texts' and major ticks' style (make the texts rotated)*/
    lv_obj_set_style_transform_rotation(scale_line, LV_SCALE_LABEL_ROTATE_MATCH_TICKS | LV_SCALE_LABEL_ROTATE_KEEP_UPRIGHT,
                                      LV_PART_INDICATOR);
    lv_obj_set_style_translate_x(scale_line, 10, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale_line, 15, LV_PART_INDICATOR);
    lv_obj_set_style_radial_offset(scale_line, 10, LV_PART_INDICATOR);

    /*Set the style of the minor ticks*/
    lv_obj_set_style_length(scale_line, 10, LV_PART_ITEMS);
    lv_obj_set_style_radial_offset(scale_line, 5, LV_PART_ITEMS);
    lv_obj_set_style_line_opa(scale_line, LV_OPA_50, LV_PART_ITEMS);

    lv_scale_set_label_show(scale_line, true);
    lv_scale_set_total_tick_count(scale_line, 31);
    lv_scale_set_major_tick_every(scale_line, 5);

    lv_scale_set_range(scale_line, 10, 40);
    lv_scale_set_angle_range(scale_line, 270);
    lv_scale_set_rotation(scale_line, 135);

    lv_obj_t * needle_line = lv_line_create(scale_line);

    lv_obj_set_style_line_width(needle_line, 3, LV_PART_MAIN);
    lv_obj_set_style_line_rounded(needle_line, true, LV_PART_MAIN);
    lv_scale_set_line_needle_value(scale_line, needle_line, 60, 33);
}

#endif
```

## Una scala orizzontale con etichette ruotate e traslate

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

/***
 * A simple horizontal scale with transforms
 */
void lv_example_scale_9(void)
{
    lv_obj_t * scale = lv_scale_create(lv_screen_active());
    lv_obj_set_size(scale, 200, 100);
    lv_scale_set_mode(scale, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_center(scale);

    lv_scale_set_label_show(scale, true);
    lv_obj_set_style_transform_rotation(scale, 450, LV_PART_INDICATOR);
    lv_obj_set_style_length(scale, 30, LV_PART_INDICATOR);
    lv_obj_set_style_translate_x(scale, 5, LV_PART_INDICATOR);

    lv_scale_set_total_tick_count(scale, 31);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 10, 40);
}

#endif
```

## Uno stile di scala circolare che simula un cardiofrequenzimetro

```
#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES && LV_FONT_MONTserrat_40 && LV_FONT_MONTserrat_18

static int32_t hr_value = 98;
static int8_t hr_step = 1;
static lv_obj_t * needle_line = NULL;
static lv_obj_t * hr_value_label = NULL;
static lv_obj_t * bpm_label = NULL;
static lv_obj_t * scale = NULL;

typedef struct {
    lv_style_t items;
    lv_style_t indicator;
    lv_style_t main;
} section_styles_t;

static section_styles_t zone1_styles;
static section_styles_t zone2_styles;
static section_styles_t zone3_styles;
static section_styles_t zone4_styles;
static section_styles_t zone5_styles;

static lv_color_t get_hr_zone_color(int32_t hr)
```

(continues on next page)

(continua dalla pagina precedente)

```

{
    if(hr < 117) return lv_palette_main(LV_PALETTE_GREY); /* Zone 1 */
    else if(hr < 135) return lv_palette_main(LV_PALETTE_BLUE); /* Zone 2 */
    else if(hr < 158) return lv_palette_main(LV_PALETTE_GREEN); /* Zone 3 */
    else if(hr < 176) return lv_palette_main(LV_PALETTE_ORANGE); /* Zone 4 */
    else return lv_palette_main(LV_PALETTE_RED); /* Zone 5 */
}

static void hr_anim_timer_cb(lv_timer_t * timer)
{
    LV_UNUSED(timer);

    hr_value += hr_step;

    if(hr_value >= 195) {
        hr_value = 195;
        hr_step = -1;
    }
    else if(hr_value <= 98) {
        hr_value = 98;
        hr_step = 1;
    }

    /* Update needle */
    lv_scale_set_line_needle_value(scale, needle_line, -8, hr_value);

    /* Update HR text */
    lv_label_set_text_fmt(hr_value_label, "%d", hr_value);

    /* Update text color based on zone */
    lv_color_t zone_color = get_hr_zone_color(hr_value);
    lv_obj_set_style_text_color(hr_value_label, zone_color, 0);
    lv_obj_set_style_text_color(bpm_label, zone_color, 0);
}

static void init_section_styles(section_styles_t * styles, lv_color_t color)
{
    lv_style_init(&styles->items);
    lv_style_set_line_color(&styles->items, color);
    lv_style_set_line_width(&styles->items, 0);

    lv_style_init(&styles->indicator);
    lv_style_set_line_color(&styles->indicator, color);
    lv_style_set_line_width(&styles->indicator, 0);

    lv_style_init(&styles->main);
    lv_style_set_arc_color(&styles->main, color);
    lv_style_set_arc_width(&styles->main, 20);
}

static void add_section(lv_obj_t * target_scale,
                      int32_t from,
                      int32_t to,
                      const section_styles_t * styles)
{
    lv_scale_section_t * sec = lv_scale_add_section(target_scale);
    lv_scale_set_section_range(target_scale, sec, from, to);
    lv_scale_set_section_style_items(target_scale, sec, &styles->items);
    lv_scale_set_section_style_indicator(target_scale, sec, &styles->indicator);
    lv_scale_set_section_style_main(target_scale, sec, &styles->main);
}

void lv_example_scale_10(void)
{
    scale = lv_scale_create(lv_screen_active());
    lv_obj_center(scale);
    lv_obj_set_size(scale, 200, 200);

    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_scale_set_range(scale, 98, 195);
    lv_scale_set_total_tick_count(scale, 15);
    lv_scale_set_major_tick_every(scale, 3);
    lv_scale_set_angle_range(scale, 280);
    lv_scale_set_rotation(scale, 130);
    lv_scale_set_label_show(scale, false);

    lv_obj_set_style_length(scale, 6, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_obj_set_style_arc_width(scale, 0, LV_PART_MAIN);

    /* Zone 1: (Grey) */
    init_section_styles(&zone1_styles, lv_palette_main(LV_PALETTE_GREY));
    add_section(scale, 98, 117, &zone1_styles);

    /* Zone 2: (Blue) */
    init_section_styles(&zone2_styles, lv_palette_main(LV_PALETTE_BLUE));
    add_section(scale, 117, 135, &zone2_styles);

    /* Zone 3: (Green) */
    init_section_styles(&zone3_styles, lv_palette_main(LV_PALETTE_GREEN));
    add_section(scale, 135, 158, &zone3_styles);

    /* Zone 4: (Orange) */
    init_section_styles(&zone4_styles, lv_palette_main(LV_PALETTE_ORANGE));
    add_section(scale, 158, 176, &zone4_styles);

    /* Zone 5: (Red) */
    init_section_styles(&zone5_styles, lv_palette_main(LV_PALETTE_RED));
    add_section(scale, 176, 195, &zone5_styles);

    needle_line = lv_line_create(scale);

    /* Optional styling */
    lv_obj_set_style_line_color(needle_line, lv_color_black(), LV_PART_MAIN);
    lv_obj_set_style_line_width(needle_line, 12, LV_PART_MAIN);
    lv_obj_set_style_length(needle_line, 20, LV_PART_MAIN);
    lv_obj_set_style_line_rounded(needle_line, false, LV_PART_MAIN);
    lv_obj_set_style_pad_right(needle_line, 50, LV_PART_MAIN);

    int32_t current_hr = 145;

    lv_scale_set_line_needle_value(scale, needle_line, 0, current_hr);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * circle = lv_obj_create(lv_scr_act());
lv_obj_set_size(circle, 130, 130);
lv_obj_center(circle);

lv_obj_set_style_radius(circle, LV_RADIUS_CIRCLE, 0);

lv_obj_set_style_bg_color(circle, lv_obj_get_style_bg_color(lv_scr_act(), LV_PART_MAIN), 0);
lv_obj_set_style_bg_opa(circle, LV_OPA_COVER, 0);
lv_obj_set_style_border_width(circle, 0, LV_PART_MAIN);

lv_obj_t * hr_container = lv_obj_create(circle);
lv_obj_center(hr_container);
lv_obj_set_size(hr_container, lv_pct(100), LV_SIZE_CONTENT);
lv_obj_set_style_bg_opa(hr_container, LV_OPA_TRANSPI, 0);
lv_obj_set_style_border_width(hr_container, 0, 0);
lv_obj_set_layout(hr_container, LV_LAYOUT_FLEX);
lv_obj_set_flex_flow(hr_container, LV_FLEX_FLOW_COLUMN);
lv_obj_set_style_pad_all(hr_container, 0, LV_PART_MAIN);
lv_obj_set_style_pad_row(hr_container, 0, 0);
lv_obj_set_flex_align(hr_container, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

hr_value_label = lv_label_create(hr_container);
lv_label_set_text_fmt(hr_value_label, "%d", current_hr);
lv_obj_set_style_text_font(hr_value_label, &lv_font_montserrat_40, 0);
lv_obj_set_style_text_align(hr_value_label, LV_TEXT_ALIGN_CENTER, 0);

bpm_label = lv_label_create(hr_container);
lv_label_set_text(bpm_label, "bpm");
lv_obj_set_style_text_font(bpm_label, &lv_font_montserrat_18, 0);
lv_obj_set_style_text_align(bpm_label, LV_TEXT_ALIGN_CENTER, 0);

lv_color_t zone_color = get_hr_zone_color(current_hr);
lv_obj_set_style_text_color(hr_value_label, zone_color, 0);
lv_obj_set_style_text_color(bpm_label, zone_color, 0);

lv_timer_create(hr_anim_timer_cb, 80, NULL);
}

#endif

```

## Uno stile di scala circolare che simula un widget tramonto/alba

```

#include "../../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES && LV_FONT_MONTserrat_12 && LV_FONT_MONTserrat_14 && LV_FONT_MONTserrat_16 && LV_FONT_MONTserrat_20

static void label_color_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    if(!draw_task) return;

    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_event_get_draw_dsc(draw_task);
    if(!base_dsc || base_dsc->part != LV_PART_INDICATOR) return;

    lv_draw_label_dsc_t * label_dsc = lv_event_get_label_dsc(draw_task);
    if(!label_dsc || !label_dsc->text) return;

    const char * txt = label_dsc->text;

    if(lv_strcmp(txt, "06") == 0 || lv_strcmp(txt, "12") == 0 ||
       lv_strcmp(txt, "18") == 0 || lv_strcmp(txt, "24") == 0) {
        label_dsc->color = lv_color_white();
    }
    else {
        label_dsc->color = lv_palette_darken(LV_PALETTE_GREY, 1);
    }
}

void lv_example_scale_11(void)
{
    lv_obj_t * bg = lv_obj_create(lv_screen_active());
    lv_obj_set_size(bg, 210, 210);
    lv_obj_center(bg);
    lv_obj_set_style_radius(bg, LV_RADIUS_CIRCLE, 0);
    lv_obj_set_style_bg_color(bg, lv_palette_darken(LV_PALETTE_GREY, 4), 0);
    lv_obj_set_style_bg_opa(bg, LV_OPA_COVER, 0);
    lv_obj_remove_flag(bg, LV_OBJ_FLAG_SCROLLABLE);
    lv_obj_set_style_pad_all(bg, 0, LV_PART_MAIN);

    lv_obj_t * scale = lv_scale_create(bg);
    lv_obj_center(scale);
    lv_obj_set_size(scale, 150, 150);
    lv_obj_set_style_arc_width(scale, 5, LV_PART_MAIN);

    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_OUTER);
    lv_scale_set_range(scale, 0, 24);
    lv_scale_set_total_tick_count(scale, 25);
    lv_scale_set_major_tick_every(scale, 1);
    lv_scale_set_angle_range(scale, 360);
    lv_scale_set_rotation(scale, 105);
    lv_scale_set_label_show(scale, true);
    lv_obj_set_style_text_font(scale, &lv_font_montserrat_12, LV_PART_INDICATOR);
    lv_obj_set_style_pad_radial(scale, -6, LV_PART_INDICATOR);

    /* Rotate the labels of the ticks */
    lv_obj_set_style_transform_rotation(scale, LV_SCALE_LABEL_ROTATE_MATCH_TICKS | LV_SCALE_LABEL_ROTATE_KEEP_UPRIGHT,
                                      LV_PART_INDICATOR);

    /* Style for major ticks */
    static lv_style_t style_ticks;
    lv_style_init(&style_ticks);
    lv_style_set_line_color(&style_ticks, lv_palette_darken(LV_PALETTE_GREY, 1));
    lv_style_set_line_width(&style_ticks, 2);
    lv_style_set_width(&style_ticks, 10);
    lv_obj_add_style(scale, &style_ticks, LV_PART_INDICATOR);

    /* Style for NIGHT - blue */
    static lv_style_t style_night;

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_init(&style_night);
lv_style_set_arc_color(&style_night, lv_palette_main(LV_PALETTE_BLUE));

/* Style for DAY – dark yellow */
static lv_style_t style_day;
lv_style_init(&style_day);
lv_style_set_arc_color(&style_day, lv_palette_darker(LV_PALETTE_YELLOW, 3));

/* NIGHT section */
lv_scale_section_t * section_night1 = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section_night1, 17, 5);
lv_scale_set_section_style_main(scale, section_night1, &style_night);

/* DAY section */
lv_scale_section_t * section_day = lv_scale_add_section(scale);
lv_scale_set_section_range(scale, section_day, 5, 17);
lv_scale_set_section_style_main(scale, section_day, &style_day);

static const char * hour_labels[] = {
    "01", "02", "03", "04", "05",
    "06", "07", "08", "09", "10",
    "11", "12", "13", "14", "15",
    "16", "17", "18", "19", "20",
    "21", "22", "23", "24",
    NULL
};
lv_scale_set_text_src(scale, hour_labels);

lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
lv_obj_add_event_cb(scale, label_color_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);

lv_obj_t * today = lv_label_create(bg);
lv_label_set_text(today, "TODAY");
lv_obj_set_style_text_font(today, &lv_font_montserrat_16, 0);
lv_obj_set_style_text_color(today, lv_color_white(), 0);
lv_obj_align(today, LV_ALIGN_TOP_MID, 0, 60);

lv_obj_t * sunrise_lbl = lv_label_create(bg);
lv_label_set_text(sunrise_lbl, "SUNRISE");
lv_obj_set_style_text_font(sunrise_lbl, &lv_font_montserrat_14, 0);
lv_obj_set_style_text_color(sunrise_lbl, lv_palette_main(LV_PALETTE_GREY), 0);
lv_obj_align(sunrise_lbl, LV_ALIGN_LEFT_MID, 37, -10);

lv_obj_t * sunrise_time = lv_label_create(bg);
lv_label_set_text(sunrise_time, "6:43");
lv_obj_set_style_text_font(sunrise_time, &lv_font_montserrat_20, 0);
lv_obj_set_style_text_color(sunrise_time, lv_color_white(), 0);
lv_obj_align_to(sunrise_time, sunrise_lbl, LV_ALIGN_OUT_BOTTOM_MID, 0, 2);

lv_obj_t * sunset_lbl = lv_label_create(bg);
lv_label_set_text(sunset_lbl, "SUNSET");
lv_obj_set_style_text_font(sunset_lbl, &lv_font_montserrat_14, 0);
lv_obj_set_style_text_color(sunset_lbl, lv_palette_main(LV_PALETTE_GREY), 0);
lv_obj_align(sunset_lbl, LV_ALIGN_RIGHT_MID, -37, -10);

lv_obj_t * sunset_time = lv_label_create(bg);
lv_label_set_text(sunset_time, "17:37");
lv_obj_set_style_text_font(sunset_time, &lv_font_montserrat_20, 0);
lv_obj_set_style_text_color(sunset_time, lv_color_white(), 0);
lv_obj_align_to(sunset_time, sunset_lbl, LV_ALIGN_OUT_BOTTOM_MID, 0, 2);
}

#endif

```

## Uno stile di scala circolare che simula una bussola

```

#include "../lv_examples.h"
#if LV_USE_SCALE && LV_BUILD_EXAMPLES

static lv_obj_t * scale;
static lv_obj_t * label;

static const char * heading_to_cardinal(int32_t heading)
{
    /* Normalize heading to range [0, 360) */
    while(heading < 0) heading += 360;
    while(heading >= 360) heading -= 360;

    if(heading < 23) return "N";
    else if(heading < 68) return "NE";
    else if(heading < 113) return "E";
    else if(heading < 158) return "SE";
    else if(heading < 203) return "S";
    else if(heading < 248) return "SW";
    else if(heading < 293) return "W";
    else if(heading < 338) return "NW";

    return "N";
}

static void set_heading_value(void * obj, int32_t v)
{
    LV_UNUSED(obj);
    lv_scale_set_rotation(scale, 270 - v);
    lv_label_set_text_fmt(label, "%d\n%s", v, heading_to_cardinal(v));
}

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    lv_draw_label_dsc_t * label_dsc = lv_draw_task_get_label_dsc(draw_task);
    lv_draw_line_dsc_t * line_dsc = lv_draw_task_get_line_dsc(draw_task);
    if(base_dsc->part == LV_PART_INDICATOR) {
        if(label_dsc) {
            if(base_dsc->id1 == 0) {
                label_dsc->color = lv_palette_main(LV_PALETTE_RED);
            }
        }
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

        }
    }
    if(line_draw_dsc) {
        if(base_dsc->id1 == 60) {
            line_draw_dsc->color = lv_palette_main(LV_PALETTE_RED);
        }
    }
}

/**
 * A round scale style simulating a compass
 */
void lv_example_scale_12(void)
{
    scale = lv_scale_create(lv_screen_active());

    lv_obj_set_size(scale, 200, 200);
    lv_scale_set_mode(scale, LV_SCALE_MODE_ROUND_INNER);
    lv_obj_set_align(scale, LV_ALIGN_CENTER);

    lv_scale_set_total_tick_count(scale, 61);
    lv_scale_set_major_tick_every(scale, 5);

    lv_obj_set_style_length(scale, 5, LV_PART_ITEMS);
    lv_obj_set_style_length(scale, 10, LV_PART_INDICATOR);
    lv_obj_set_style_line_width(scale, 3, LV_PART_INDICATOR);
    lv_scale_set_range(scale, 0, 360);

    static const char * custom_labels[] = {"N", "30", "60", "E", "120", "150", "S", "210", "240", "W", "300", "330", NULL};
    lv_scale_set_text_src(scale, custom_labels);

    lv_scale_set_angle_range(scale, 360);
    lv_scale_set_rotation(scale, 270);

    lv_obj_add_event_cb(scale, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(scale, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, 100);
    lv_obj_set_align(label, LV_ALIGN_CENTER);
    lv_label_set_text(label, "\u00d8\nN");
    lv_obj_set_style_text_align(label, LV_TEXT_ALIGN_CENTER, 0);

    set_heading_value(NULL, 0);

    lv_obj_t * symbol = lv_label_create(scale);
    lv_obj_set_align(symbol, LV_ALIGN_TOP_MID);
    lv_obj_set_y(symbol, 5);
    lv_label_set_text(symbol, LV_SYMBOL_UP);
    lv_obj_set_style_text_align(symbol, LV_TEXT_ALIGN_CENTER, 0);
    lv_obj_set_style_text_color(symbol, lv_palette_main(LV_PALETTE_RED), 0);

    lv_anim_t anim_scale;
    lv_anim_init(&anim_scale);
    lv_anim_set_var(&anim_scale, scale);
    lv_anim_set_exec_cb(&anim_scale, set_heading_value);
    lv_anim_set_duration(&anim_scale, 5000);
    lv_anim_set_repeat_delay(&anim_scale, 500);
    lv_anim_set_repeat_count(&anim_scale, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_reverse_duration(&anim_scale, 5000);
    lv_anim_set_reverse_delay(&anim_scale, 500);
    lv_anim_set_values(&anim_scale, 0, 360);
    lv_anim_start(&anim_scale);
}
#endif

```

## Segni di spunta e etichette degli assi con scorrimento su un grafico

```

#include "../../lv_examples.h"
#if LV_USE_CHART && LV_BUILD_EXAMPLES

/***
 * Use lv_scale to add ticks to a scrollable chart
 */
void lv_example_chart_2(void)
{
    /*Create a container*/
    lv_obj_t * main_cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(main_cont, 200, 150);
    lv_obj_center(main_cont);

    /*Create a transparent wrapper for the chart and the scale.
     *Set a large width, to make it scrollable on the main container*/
    lv_obj_t * wrapper = lv_obj_create(main_cont);
    lv_obj_remove_style_all(wrapper);
    lv_obj_set_size(wrapper, lv_pct(300), lv_pct(100));
    lv_obj_set_flex_flow(wrapper, LV_FLEX_FLOW_COLUMN);

    /*Create a chart on the wrapper
     *Set it's width to 100% to fill the large wrapper*/
    lv_obj_t * chart = lv_chart_create(wrapper);
    lv_obj_set_width(chart, lv_pct(100));
    lv_obj_set_flex_grow(chart, 1);
    lv_chart_set_type(chart, LV_CHART_TYPE_BAR);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_PRIMARY_Y, 0, 100);
    lv_chart_set_axis_range(chart, LV_CHART_AXIS_SECONDARY_Y, 0, 400);
    lv_chart_set_point_count(chart, 12);
    lv_obj_set_style_radius(chart, 0, 0);

    /*Create a scale also with 100% width*/
    lv_obj_t * scale_bottom = lv_scale_create(wrapper);
    lv_scale_set_mode(scale_bottom, LV_SCALE_MODE_HORIZONTAL_BOTTOM);
    lv_obj_set_size(scale_bottom, lv_pct(100), 25);
    lv_scale_set_total_tick_count(scale_bottom, 12);
    lv_scale_set_major_tick_every(scale_bottom, 1);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_style_pad_hor(scale_bottom, lv_chart_get_first_point_center_offset(chart), 0);
static const char * month[] = {"Jan", "Febr", "March", "Apr", "May", "Jun", "July", "Aug", "Sept", "Oct", "Nov", "Dec", NULL};
lv_scale_set_text_src(scale_bottom, month);

/*Add two data series*/
lv_chart_series_t * ser1 = lv_chart_add_series(chart, lv_palette_lighten(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);
lv_chart_series_t * ser2 = lv_chart_add_series(chart, lv_palette_darken(LV_PALETTE_GREEN, 2), LV_CHART_AXIS_PRIMARY_Y);

/*Set the next points on 'ser1'*/
uint32_t i;
for(i = 0; i < 12; i++) {
    lv_chart_set_next_value(chart, ser1, (int32_t)lv_rand(10, 60));
    lv_chart_set_next_value(chart, ser2, (int32_t)lv_rand(50, 90));
}
lv_chart_refresh(chart); /*Required after direct set*/
}

#endif

```

## 6.25.7 API

*Scale (lv\_scale)* .. Autogenerato

*lv\_scale.h*

*lv\_scale\_private.h*

## 6.26 Slider (lv\_slider)

### 6.26.1 Panoramica

Il Widget Slider ha l'aspetto di una *Bar (lv\_bar)* con una manopola. La manopola può essere trascinata per impostare il valore dello Slider. Come la Barra, uno Slider può essere verticale o orizzontale.

### 6.26.2 Parti e Stili

- *LV\_PART\_MAIN* Il background dello Slider. Utilizza le *tipiche proprietà di stile del background*. *padding* rende l'indicatore più piccolo nella rispettiva direzione.
- *LV\_PART\_INDICATOR* L'indicatore che mostra lo stato corrente dello Slider; utilizza anche le *tipiche proprietà di stile del background*.
- *LV\_PART\_KNOB* Un rettangolo (o cerchio) disegnato in corrispondenza del valore corrente; utilizza anche le *tipiche proprietà di stile del background* per descrivere la/le manopola/e. Per default, la manopola è rotonda (lo stile del raggio può modificarlo) con una lunghezza del lato pari alla dimensione minore dello Slider. La manopola può essere ingrandita con i valori di *padding*. I valori di padding possono anche essere asimmetrici.

### 6.26.3 Utilizzo

#### Valore, intervallo e orientamento

Una volta creato, uno Slider ha:

- valore == 0
- intervallo di default di [0..100],
- orientamento orizzontale, con
- larghezza di default di circa 2 pollici (in base al valore configurato di *LV\_DPI\_DEF*),
- altezza di default di circa 1/10 di pollice (in base al valore configurato di *LV\_DPI\_DEF*).

Per impostare valori diversi, utilizzare:

- *lv\_slider\_set\_value*(slider, new\_value, LV\_ANIM\_ON / OFF) (il tempo di animazione è impostato dalla proprietà *anim\_time* degli stili);
- *lv\_slider\_set\_range*(slider, min, max); e

- per orientamento, larghezza e altezza, è sufficiente impostare le proprietà width e height;
- `lv_slider_set_orientation(slider, orientation)` per sovrascrivere l'orientamento determinato da width e height. I valori validi per orientation sono:

- `LV_SLIDER_ORIENTATION_AUTO`
- `LV_SLIDER_ORIENTATION_HORIZONTAL`
- `LV_SLIDER_ORIENTATION_VERTICAL`

La direzione di disegno di default è da sinistra a destra nell'orientamento orizzontale e dal basso verso l'alto nell'orientamento verticale. Se il valore minimo è impostato su un valore maggiore del valore massimo (ad esempio [100..0]), la direzione di disegno viene invertita.

## Modes

Lo Slider può essere in una delle seguenti modalità:

- `LV_SLIDER_MODE_NORMAL` Uno Slider normale come descritto sopra (il default)
- `LV_SLIDER_SYMMETRICAL` Disegna l'indicatore dal valore zero al valore corrente. Richiede un intervallo minimo negativo e un intervallo massimo positivo.
- `LV_SLIDER_RANGE` Consente anche di impostare il valore iniziale tramite `lv_bar_set_start_value(slider, new_value, LV_ANIM_ON / OFF)`. Il valore iniziale deve essere sempre inferiore al valore finale.

La modalità può essere modificata con `lv_slider_set_mode(slider, LV_SLIDER_MODE_...)`

## Modalità solo manopola

Normalmente, lo Slider può essere regolato trascinando la manopola o cliccando sulla barra dello Slider. In quest'ultimo caso, la manopola si sposta nel punto cliccato e il valore dello Slider cambia di conseguenza. In alcuni casi è preferibile impostare lo Slider in modo che reagisca solo al trascinamento della manopola. Questa funzione si attiva aggiungendo il flag `LV_OBJ_FLAG_ADV_HITTEST`: `lv_obj_add_flag(slider, LV_OBJ_FLAG_ADV_HITTEST)`.

Qualsiasi area di clic estesa (impostata da `lv_obj_set_ext_click_area(slider, value)`) aumenta l'area di clic della manopola.

### 6.26.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato mentre lo Slider viene trascinato o modificato con i tasti. L'evento viene inviato continuamente mentre lo Slider viene trascinato
- `LV_EVENT_RELEASED` inviato una volta quando lo Slider viene rilasciato.

#### Ulteriori Informazioni

*Eventi della Bar.*

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

### 6.26.5 Tasti

- `LV_KEY_UP/RIGHT` Incrementa il valore dello Slider di 1.
- `LV_KEY_DOWN/LEFT` Decrementa il valore dello Slider di 1.

## Ulteriori Informazioni

Scoprire di più sui *Tasti*.

### 6.26.6 Esempio

#### Slider semplice

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);
static lv_obj_t * slider_label;

/***
 * A default slider with a label displaying the current value
 */
void lv_example_slider_1(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

    lv_obj_set_style_anim_duration(slider, 2000, 0);
    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_screen_active());
    lv_label_set_text(slider_label, "0");

    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target_obj(e);
    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d%%", (int)lv_slider_get_value(slider));
    lv_label_set_text(slider_label, buf);
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

#endif
```

#### Slider con stile personalizzato

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

/***
 * Show how to style a slider.
 */
void lv_example_slider_2(void)
{
    /*Create a transition*/
    static const lv_style_prop_t props[] = {LV_STYLE_BG_COLOR, 0};
    static lv_style_transition_dsc_t transition_dsc;
    lv_style_transition_dsc_init(&transition_dsc, props, lv_anim_path_linear, 300, 0, NULL);

    static lv_style_t style_main;
    static lv_style_t style_indicator;
    static lv_style_t style_knob;
    static lv_style_t style_pressed_color;
    lv_style_init(&style_main);
    lv_style_set_bg_opa(&style_main, LV_OPA_COVER);
    lv_style_set_bg_color(&style_main, lv_color_hex3(0xbbb));
    lv_style_set_radius(&style_main, LV_RADIUS_CIRCLE);
    lv_style_set_pad_ver(&style_main, -2); /*Makes the indicator larger*/

    lv_style_init(&style_indicator);
    lv_style_set_bg_opa(&style_indicator, LV_OPA_COVER);
    lv_style_set_bg_color(&style_indicator, lv_palette_main(LV_PALETTE_CYAN));
    lv_style_set_radius(&style_indicator, LV_RADIUS_CIRCLE);
    lv_style_set_transition(&style_indicator, &transition_dsc);

    lv_style_init(&style_knob);
    lv_style_set_bg_opa(&style_knob, LV_OPA_COVER);
    lv_style_set_bg_color(&style_knob, lv_palette_main(LV_PALETTE_CYAN));
    lv_style_set_border_color(&style_knob, lv_palette_darken(LV_PALETTE_CYAN, 3));
    lv_style_set_border_width(&style_knob, 2);
    lv_style_set_radius(&style_knob, LV_RADIUS_CIRCLE);
    lv_style_set_pad_all(&style_knob, 6); /*Makes the knob larger*/
    lv_style_set_transition(&style_knob, &transition_dsc);

    lv_style_init(&style_pressed_color);
    lv_style_set_bg_color(&style_pressed_color, lv_palette_darken(LV_PALETTE_CYAN, 2));

    /*Create a slider and add the style*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_remove_style_all(slider); /*Remove the styles coming from the theme*/

    lv_obj_add_style(slider, &style_main, LV_PART_MAIN);
    lv_obj_add_style(slider, &style_indicator, LV_PART_INDICATOR);
    lv_obj_add_style(slider, &style_pressed_color, LV_PART_INDICATOR | LV_STATE_PRESSED);
    lv_obj_add_style(slider, &style_knob, LV_PART_KNOB);
    lv_obj_add_style(slider, &style_pressed_color, LV_PART_KNOB | LV_STATE_PRESSED);

    lv_obj_center(slider);
}
```

## Slider estensibile

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

#define MAX_VALUE 100
#define MIN_VALUE 0

static void slider_event_cb(lv_event_t * e);

/**
 * Show the current value when the slider is pressed by extending the drawer
 *
 */
void lv_example_slider_3(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider;
    slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);

    lv_slider_set_mode(slider, LV_SLIDER_MODE_RANGE);
    lv_slider_set_range(slider, MIN_VALUE, MAX_VALUE);
    lv_slider_set_value(slider, 70, LV_ANIM_OFF);
    lv_slider_set_start_value(slider, 20, LV_ANIM_OFF);

    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_refresh_ext_draw_size(slider);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);

    /*Provide some extra space for the value*/
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        lv_event_set_ext_draw_size(e, 50);
    }
    else if(code == LV_EVENT_DRAW_MAIN_END) {
        if(!lv_obj_has_state(obj, LV_STATE_PRESSED)) return;

        lv_area_t slider_area;
        lv_obj_get_coords(obj, &slider_area);
        lv_area_t indic_area = slider_area;
        lv_area_set_width(&indic_area, lv_area_get_width(&slider_area) * lv_slider_get_value(obj) / MAX_VALUE);
        indic_area.x1 += lv_area_get_width(&slider_area) * lv_slider_get_left_value(obj) / MAX_VALUE;
        char buf[16];
        lv_snprintf(buf, sizeof(buf), "%d - %d", (int)lv_slider_get_left_value(obj), (int)lv_slider_get_value(obj));

        lv_text_attributes_t attributes = {};
        attributes.text_flags = LV_TEXT_FLAG_NONE;
        attributes.max_width = LV_COORD_MAX;
        attributes.letter_space = 0;
        attributes.line_space = 0;

        lv_point_t label_size;
        lv_text_get_size(&label_size, buf, LV_FONT_DEFAULT, &attributes);
        lv_area_t label_area;
        label_area.x1 = 0;
        label_area.x2 = label_size.x - 1;
        label_area.y1 = 0;
        label_area.y2 = label_size.y - 1;

        lv_area_align(&indic_area, &label_area, LV_ALIGN_OUT_TOP_MID, 0, -10);

        lv_draw_label_dsc_t label_draw_dsc;
        lv_draw_label_dsc_init(&label_draw_dsc);
        label_draw_dsc.color = lv_color_hex3(0x888);
        label_draw_dsc.text = buf;
        label_draw_dsc.text_local = true;
        lv_layer_t * layer = lv_event_get_layer(e);
        lv_draw_label(layer, &label_draw_dsc, &label_area);
    }
}
#endif
```

## Slider con direzione opposta

```
#include "../../lv_examples.h"
#if LV_USE_SLIDER && LV_BUILD_EXAMPLES

static void slider_event_cb(lv_event_t * e);
static lv_obj_t * slider_label;

/**
 * Slider with opposite direction
 */
void lv_example_slider_4(void)
{
    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    /*Reverse the direction of the slider*/
    lv_slider_set_range(slider, 100, 0);
    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_screen_active());
    lv_label_set_text(slider_label, "0");

    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}

static void slider_event_cb(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_target_obj(e);
    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d%", (int)lv_slider_get_value(slider));
```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_label_set_text(slider_label, buf);
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
}
#endif

```

## 6.26.7 API

*lv\_api\_map\_v9\_1.h*

*lv\_observer.h*

*lv\_obj\_property\_names.h*

*lv\_slider.h*

*lv\_slider\_private.h*

## 6.27 Spangroup (*lv\_spangroup*)

### 6.27.1 Panoramica

Il Widget Spangroup viene utilizzato per visualizzare testo formattato. A differenza del Widget Label, gli Spangroup possono visualizzare testo con stili diversi, con colori e dimensioni diversi. Vedere esempio di seguito.

Uno Spangroup contiene 0 o più Descrittori Span ("Span"). Ogni Span contiene il proprio testo e le proprie proprietà di stile. Si aggiunge 1 Span (come elemento figlio) allo Spangroup per ogni "span" di testo con stile univoco necessario. Ogni Span aggiunto viene posto alla fine dell'elenco. La sequenza dell'elenco determina l'ordine in cui gli Span vengono visualizzati. Gli Span possono essere aggiunti e rimossi dallo Spangroup per tutta la sua durata. Il numero di Span che possono essere aggiunti è limitato solo dalla RAM disponibile.

### 6.27.2 Parti e Stili

- *LV\_PART\_MAIN* Lo Spangroup ha una sola parte.

### 6.27.3 Utilizzo

#### Impostare testo e stile

Ogni Span necessario a uno Spangroup si aggiunge in questo modo:

```
lv_span_t * span = lv_spangroup_new_span(spangroup);
```

Dopo aver creato uno Span, si usano le seguenti funzioni per impostarne le proprietà di testo e stile:

- *lv\_span\_set\_text*(span, "text")
- *lv\_style\_set\_<property\_name>*(&span->style, value)

Esempio di quest'ultimo: *lv\_style\_set\_text\_color*(&span->style, *lv\_palette\_main(LV\_PALETTE\_RED)*).

Se *mode* != *LV\_SPAN\_MODE\_FIXED* del Widget Spangroup, chiamare *lv\_spangroup\_refr\_mode*(spangroup) dopo aver modificato uno qualsiasi dei suoi Span per assicurarsi che venga ridisegnato correttamente.

#### Recupero di uno Span figlio

Gli Spangroup memorizzano i propri elementi figlio in modo diverso dai normali Widget, quindi le normali funzioni per ottenere elementi figlio non funzioneranno.

*lv\_spangroup\_get\_child*(spangroup, id) restituirà un puntatore all'elemento figlio Span all'indice *id*. Inoltre, *id* può essere negativo per indicizzare dalla fine dello Spangroup, dove -1 è l'elemento figlio più giovane, -2 è il secondo più giovane, ecc.

Ad esempio `lv_span_t * span = lv_spangroup_get_child(spangroup, 0)` restituirà il primo elemento figlio dello Spangroup. `lv_span_t * span = lv_spangroup_get_child(spangroup, -1)` restituirà l'ultimo elemento figlio (o il più recente).

### Conteggio dei figli

Su usa `lv_spangroup_get_span_count(spangroup)` per ottenere il numero di Span contenuti.

Ad esempio `uint32_t size = lv_spangroup_get_span_count(spangroup)`

### Rimozione di uno Span

È possibile rimuovere uno Span in qualsiasi momento durante la vita dello Spangroup utilizzando la funzione `lv_spangroup_delete_span(spangroup, span)`.

### Allineamento del testo

Come il Widget Label, uno Spangroup può essere impostato su una delle seguenti modalità di allineamento del testo:

- `LV_TEXT_ALIGN_LEFT` Allinea il testo a sinistra.
- `LV_TEXT_ALIGN_CENTER` Centra il testo.
- `LV_TEXT_ALIGN_RIGHT` Allinea il testo al bordo destro.
- `LV_TEXT_ALIGN_AUTO` Allinea automaticamente.

Usare la funzione `lv_spangroup_set_align(spangroup, LV_TEXT_ALIGN_...)` per impostare l'allineamento del testo.

### Modes

Uno Spangroup può essere impostato su una delle seguenti modalità:

- `LV_SPAN_MODE_FIXED` Ne fissa le dimensioni.
- `LV_SPAN_MODE_EXPAND` Espande le dimensioni fino a raggiungere quelle del testo, ma rimane su una sola riga.
- `LV_SPAN_MODE_BREAK` Mantiene la larghezza; interrompe le righe troppo lunghe ed espande automaticamente l'altezza.

Usare `lv_spangroup_set_mode(spangroup, LV_SPAN_MODE_BREAK)` per impostarne la modalità.

### Overflow

Uno Spangroup può essere impostato per gestire l'overflow del testo in uno dei seguenti modi:

- `LV_SPAN_OVERFLOW_CLIP` tronca il testo al limite dell'area.
- `LV_SPAN_OVERFLOW_ELLIPSIS` visualizza un'ellissi (...) quando il testo supera l'area.

Use `lv_spangroup_set_overflow(spangroup, LV_SPAN_OVERFLOW_CLIP)` per impostare la modalità di overflow dello Spangroup.

### Rientro prima riga

Usare `lv_spangroup_set_indent(spangroup, 20)` per impostare il rientro della prima riga. Tutte le modalità supportano le unità pixel. Inoltre, le modalità `LV_SPAN_MODE_FIXED` e `LV_SPAN_MODE_BREAK` supportano `unità percentuali`. anche.

## Righe

Usare `lv_spangroup_set_max_lines(spangroup, 10)` per impostare il numero massimo di righe da visualizzare in modalità `LV_SPAN_MODE_BREAK`. Un valore negativo indica nessun limite.

### 6.27.4 Eventi

I Widget Span non inviano eventi speciali.

#### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

### 6.27.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Span.

#### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

### 6.27.6 Esempio

#### Span con stili personalizzati

```
#include "../../../lv_examples.h"
#if LV_USE_SPAN && LV_BUILD_EXAMPLES

static void click_event_cb(lv_event_t * e)
{
    lv_indev_t * indev = lv_event_get_indev(e);
    lv_point_t point;
    lv_indev_get_point(indev, &point);
#if LV_USE_LOG
    lv_obj_t * spans = lv_event_get_target_obj(e);
    lv_span_t * span = lv_spangroup_get_span_by_point(spans, &point);
    LV_LOG_USER("%s", span ? lv_span_get_text(span) : "NULL");
#endif
}

/**
 * Create spans and get clicked one
 */
void lv_example_span_1(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_border_width(&style, 1);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_ORANGE));
    lv_style_set_pad_all(&style, 2);

    lv_obj_t * spans = lv_spangroup_create(lv_screen_active());
    /* Setting a fixed width and height to LV_SIZE_CONTENT will make the text wrap */
    lv_obj_set_width(spans, 300);
    lv_obj_set_height(spans, LV_SIZE_CONTENT);
    lv_obj_center(spans);
    lv_obj_add_style(spans, &style, 0);
    lv_obj_add_flag(spans, LV_OBJ_FLAG_CLICKABLE);

    lv_spangroup_set_align(spans, LV_TEXT_ALIGN_LEFT);
    lv_spangroup_set_overflow(spans, LV_SPAN_OVERFLOW_CLIP);
    lv_spangroup_set_indent(spans, 20);

    lv_span_t * span = lv_spangroup_add_span(spans);
    lv_span_set_text(span, "China is a beautiful country.");
    lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_RED));
    lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_UNDERLINE);
    lv_style_set_text_opa(lv_span_get_style(span), LV_OPA_50);

    span = lv_spangroup_add_span(spans);
    lv_span_set_text_static(span, "good good study, day day up.");
#if LV_FONT_MONTserrat_24
    lv_style_set_text_font(lv_span_get_style(span), &lv_font_montserrat_24);
#endif
    lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_GREEN));

    span = lv_spangroup_add_span(spans);
    lv_span_set_text_static(span, "LVGL is an open-source graphics library.");
    lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_BLUE));

    span = lv_spangroup_add_span(spans);
    lv_span_set_text_static(span, "the boy no name.");
    lv_style_set_text_color(lv_span_get_style(span), lv_palette_main(LV_PALETTE_GREEN));
#endif
    if LV_FONT_MONTserrat_20

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_text_font(lv_span_get_style(span), &lv_font_montserrat_20);
#endif
lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_UNDERLINE);

span = lv_spangroup_add_span(spans);
lv_span_set_text(span, "I have a dream that hope to come true.");
lv_style_set_text_decor(lv_span_get_style(span), LV_TEXT_DECOR_STRIKETHROUGH);

lv_spangroup_refresh(spans);

lv_obj_add_event_cb(spans, click_event_cb, LV_EVENT_CLICKED, NULL);
}

#endif

```

## 6.27.7 API

*lv\_api\_map\_v9\_1.h*

*lv\_span.h*

*lv\_span\_private.h*

## 6.28 Spinbox (lv\_spinbox)

### 6.28.1 Panoramica

Spinbox contiene un numero intero visualizzato come numero decimale con una possibile posizione fissa del punto decimale e un numero di cifre configurabile. Il valore può essere aumentato o diminuito tramite *Tasti* o funzioni API. Spinbox è una *Text Area* (*lv\_textarea*) con comportamenti estesi per consentire la visualizzazione e la modifica di un valore numerico con vincoli configurabili.

### 6.28.2 Parti e Stili

Le parti di Spinbox sono identiche a quelle della *Text Area*.

#### Valore, intervallo e passo

- *lv\_spinbox\_set\_value*(spinbox, 1234) imposta un nuovo valore per lo Spinbox.
- *lv\_spinbox\_increment*(spinbox) e *lv\_spinbox\_decrement*(spinbox) incrementano/decrementano il valore dello Spinbox in base alla cifra attualmente selezionata.
- *lv\_spinbox\_set\_range*(spinbox, -1000, 2500) ne imposta l'intervallo. Se il valore viene modificato da *lv\_spinbox\_set\_value*(spinbox), dai *Tasti*, da *lv\_spinbox\_increment*(spinbox) o da *lv\_spinbox\_decrement*(spinbox) questo intervallo verrà rispettato.
- *lv\_spinbox\_set\_step*(spinbox, 100) imposta quale cifra modificare durante l'incremento/decremento. È possibile impostare solo multipli di dieci.
- *lv\_spinbox\_set\_cursor\_pos*(spinbox, 1) imposta il cursore su una cifra specifica da modificare in caso di incremento/decremento. La posizione '0' imposta il cursore sulla cifra meno significativa.

Se si utilizza un encoder come dispositivo di input, la cifra selezionata viene spostata a destra per default ogni volta che si clicca sul pulsante dell'encoder. Per modificare questo comportamento e spostarla a sinistra, è possibile utilizzare *lv\_spinbox\_set\_digit\_step\_direction*(spinbox, *LV\_DIR\_LEFT*) can be used.

#### Formato

*lv\_spinbox\_set\_digit\_format*(spinbox, digit\_count, separator\_position) imposta il formato del numero. *digit\_count* è il numero totale di cifre da visualizzare. *separator\_position* è il numero di cifre iniziali prima della virgola. Impostare 0 per *separator\_position* per non visualizzare alcun separatore decimale.

## Rollover

`lv_spinbox_set_rollover(spinbox, true / false)` abilita/disabilita la modalità rollover. Se viene raggiunto il valore minimo o massimo con il rollover abilitato e l'utente tenta di continuare a modificare il valore nella stessa direzione, il valore verrà modificato fino all'altro limite. Se il rollover è disabilitato, il valore si fermerà al valore minimo o massimo.

### 6.28.3 Eventi

- `LV_EVENT_VALUE_CHANGED` Invia quando il valore è cambiato.

#### Ulteriori Informazioni

*Textarea Events.*

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

### 6.28.4 Tasti

- `LV_KEY_LEFT/RIGHT` Con il *tastierino numerico* sposta il cursore a sinistra/destra. Con L'*Encoder* decrementa/incrementa la cifra selezionata.
- `LV_KEY_UP/DOWN` Con *tastierino numerico* e *Encoder* incrementa/decrementa il valore.
- `LV_KEY_ENTER` con l'*Encoder*, sposta il focus sulla cifra successiva. Se il focus è sull'ultima cifra, il focus si sposta sulla prima.

#### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

### 6.28.5 Esempio

#### Spinbox semplice

```
#include "../../lv_examples.h"
#if LV_USE_SPINBOX && LV_BUILD_EXAMPLES

static lv_obj_t * spinbox;

static void lv_spinbox_increment_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_increment(spinbox);
    }
}

static void lv_spinbox_decrement_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_SHORT_CLICKED || code == LV_EVENT_LONG_PRESSED_REPEAT) {
        lv_spinbox_decrement(spinbox);
    }
}

void lv_example_spinbox_1(void)
{
    spinbox = lv_spinbox_create(lv_screen_active());
    lv_spinbox_set_range(spinbox, -1000, 25000);
    lv_spinbox_set_digit_format(spinbox, 5, 2);
    lv_spinbox_step_prev(spinbox);
    lv_obj_set_width(spinbox, 100);
    lv_obj_center(spinbox);

    int32_t h = lv_obj_get_height(spinbox);

    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_RIGHT_MID, 5, 0);
    lv_obj_set_style_bg_image_src(btn, LV_SYMBOL_PLUS, 0);
    lv_obj_add_event_cb(btn, lv_spinbox_increment_event_cb, LV_EVENT_ALL, NULL);

    btn = lv_button_create(lv_screen_active());
    lv_obj_set_size(btn, h);
    lv_obj_align_to(btn, spinbox, LV_ALIGN_OUT_LEFT_MID, -5, 0);
    lv_obj_set_style_bg_image_src(btn, LV_SYMBOL_MINUS, 0);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_obj_add_event_cb(btn, lv_spinbox_decrement_event_cb, LV_EVENT_ALL, NULL);
}

#endif

```

## 6.28.6 API

*lv\_spinbox.h*

*lv\_spinbox\_private.h*

# 6.29 Spinner (lv\_spinner)

## 6.29.1 Panoramica

Il Widget Spinner è un arco rotante lungo un anello, solitamente utilizzato per indicare che è in corso un'attività.

## 6.29.2 Parti e Stili

Le parti dello Spinner sono identiche a quelle di [Arc](#).

## 6.29.3 Utilizzo

### Creare uno Spinner

Per creare uno Spinner, si usa `lv_spinner_create(parent)`.

Usare `lv_spinner_set_anim_params(spinner, spin_duration, angle)` per personalizzare la durata di una rotazione e la lunghezza dell'arco.

## 6.29.4 Eventi

I Widget Spinner non inviano eventi speciali.

### Ulteriori Informazioni

*Eventi di Arc*

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

## 6.29.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Spinner.

### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

## 6.29.6 Esempio

### Spinner semplice

```

#include "../../lv_examples.h"
#if LV_USE_SPINNER && LV_BUILD_EXAMPLES

void lv_example_spinner_1(void)
{
    /*Create a spinner*/
    lv_obj_t * spinner = lv_spinner_create(lv_screen_active());

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_size(spinner, 100, 100);
lv_obj_center(spinner);
lv_spinner_set_anim_params(spinner, 10000, 200);
}

#endif

```

## 6.29.7 API

*lv\_spinner.h*

# 6.30 Switch (lv\_switch)

## 6.30.1 Panoramica

I Widget Switch hanno l'aspetto di piccoli cursori e vengono utilizzati per visualizzare, e facoltativamente modificare, un valore che può essere "on" o "off".

Per default, uno Switch è orientato orizzontalmente. Il suo orientamento sarà verticale se si impone `width < height`.

## 6.30.2 Parti e Stili

- *LV\_PART\_MAIN* Background dello Switch; utilizza le *tipiche proprietà di stile del background*. `padding` rende l'indicatore più piccolo nella rispettiva direzione.
- *LV\_PART\_INDICATOR* L'indicatore che mostra lo stato corrente dello Switch; utilizza anche le *tipiche proprietà di stile del background*.
- *LV\_PART\_KNOB* Un rettangolo (o cerchio) disegnato a sinistra o a destra dell'indicatore; utilizza anche le *tipiche proprietà di stile del background* per modificare l'aspetto della manopola. Per default, la manopola è rotonda (lo stile `radius` può modificarlo) con un diametro leggermente inferiore al lato più piccolo dello slider. La manopola può essere ingrandita con i valori di `padding`. I valori di padding possono anche essere asimmetrici.

## 6.30.3 Utilizzo

### Cambio di stato

Lo Switch utilizza lo stato standard *LV\_STATE\_CHECKED* state.

Per ottenere lo stato corrente dello Switch (con `true` impostato su ON), utilizzare `lv_obj_has_state(widget, LV_STATE_CHECKED)`.

Chiamare `lv_obj_add_state(widget, LV_STATE_CHECKED)` per attivarlo, oppure `lv_obj_remove_state(widget, LV_STATE_CHECKED)` per disattivarlo da programma.

### Modificare l'orientamento

Quando viene creato uno Switch, il suo orientamento predefinito è *LV\_SWITCH\_ORIENTATION\_AUTO*, che ne determina l'orientamento in base a `width` e `height`. È possibile modificare questo comportamento utilizzando `lv_switch_set_orientation(widget, orientation)`. I valori possibili per `orientation` sono:

- *LV\_SWITCH\_ORIENTATION\_AUTO*
- *LV\_SWITCH\_ORIENTATION\_HORIZONTAL*
- *LV\_SWITCH\_ORIENTATION\_VERTICAL*

## 6.30.4 Eventi

- *LV\_EVENT\_VALUE\_CHANGED* Inviato quando lo Switch cambia stato.

### **i** Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

## 6.30.5 Tasti

- LV\_KEY\_UP/RIGHT Accende lo Switch
- LV\_KEY\_DOWN/LEFT Spegne lo Switch
- *LV\_KEY\_ENTER* Attiva/disattiva lo Switch

### **i** Ulteriori Informazioni

Scoprire di più sui *Tasti*.

## 6.30.6 Esempio

### Switch semplice

```
#include "../../lv_examples.h"
#if LV_USE_SWITCH && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_UNUSED(obj);
        LV_LOG_USER("State: %s\n", lv_obj_has_state(obj, LV_STATE_CHECKED) ? "On" : "Off");
    }
}

void lv_example_switch_1(void)
{
    lv_obj_set.flex_flow(lv_screen_active(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set.flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    lv_obj_t * sw;
    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_add_flag(sw, LV_OBJ_FLAG_EVENT_BUBBLE);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

    sw = lv_switch_create(lv_screen_active());
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_state(sw, LV_STATE_DISABLED);
    lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

### Orientamento dello Switch

```
#include "../../lv_examples.h"
#if LV_USE_SWITCH && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        LV_UNUSED(obj);
        LV_LOG_USER("State: %s\n", lv_obj_has_state(obj, LV_STATE_CHECKED) ? "On" : "Off");
    }
}

void lv_example_switch_2(void)
{
    lv_obj_set.flex_flow(lv_screen_active(), LV_FLEX_FLOW_ROW);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_flex_align(lv_screen_active(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

lv_obj_t * sw;
sw = lv_switch_create(lv_screen_active());
lv_obj_set_size(sw, 30, 60);
lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);

sw = lv_switch_create(lv_screen_active());
lv_obj_set_size(sw, 30, 60);
lv_switch_set_orientation(sw, LV_SWITCH_ORIENTATION_VERTICAL);
lv_obj_add_state(sw, LV_STATE_CHECKED);
lv_obj_add_event_cb(sw, event_handler, LV_EVENT_ALL, NULL);
}

#endif

```

## 6.30.7 API

*lv\_switch.h*

*lv\_switch\_private.h*

# 6.31 Table (lv\_table)

## 6.31.1 Panoramica

Le "Table" [tabelle] sono costruite da righe, colonne e celle contenenti testo.

Il Widget Table è molto leggero perché memorizza solo le stringhe di testo. Non vengono creati veri e propri widget per le celle: vengono semplicemente disegnati al volo.

La Tabella viene aggiunta al gruppo di default (se impostato). La Table è un Widget modificabile, che consente la selezione di una cella con encoder e navigazione da tastiera.

## 6.31.2 Parti e Stili

- *LV\_PART\_MAIN* Il background della Table utilizza le *tipiche proprietà di stile del background*.
- *LV\_PART\_ITEMS* Anche le celle della tabella utilizzano le *tipiche proprietà di stile del background* e le proprietà di stile del testo.

## 6.31.3 Utilizzo

### Impostare il valore della cella

Le celle possono memorizzare solo testo, quindi i numeri devono essere convertiti in testo prima di essere visualizzati in una Tabella.

*lv\_table\_set\_cell\_value(table, row, col, "Content")*. Il testo viene salvato dalla Tabella, quindi il buffer contenente la stringa può essere una variabile locale.

È possibile utilizzare interruzioni di riga nel testo, ad esempio "Value\n60.3".

Nuove righe e colonne vengono aggiunte automaticamente in base alle esigenze.

### Righe e Colonne

Per impostare esplicitamente il numero di righe e colonne, si usano *lv\_table\_set\_row\_count*(table, row\_cnt) e *lv\_table\_set\_column\_count*(table, col\_cnt).

### Larghezza e Altezza

La larghezza delle colonne può essere impostata con *lv\_table\_set\_column\_width*(table, col\_id, width). La larghezza complessiva del Widget Table sarà impostata sulla somma delle larghezze di tutte le colonne.

L'altezza viene calcolata automaticamente in base agli stili delle celle (carattere, spaziatura interna, ecc.) e al numero di righe.

## Unione delle celle

Le celle possono essere unite orizzontalmente con `lv_table_set_cell_ctrl(table, row, col, LV_TABLE_CELL_CTRL_MERGE_RIGHT)`. Per unire più celle adiacenti, chiamare questa funzione per ogni cella.

## Scrolling

Se la larghezza o l'altezza di una Tabella è impostata su `LV_SIZE_CONTENT`, tale dimensione verrà utilizzata per visualizzare l'intera tabella nella rispettiva direzione. Ad esempio, `lv_obj_set_size(table, LV_SIZE_CONTENT, LV_SIZE_CONTENT)` imposta automaticamente la dimensione della Tabella per visualizzare tutte le colonne e le righe.

Se la larghezza o l'altezza sono impostate su un numero inferiore alla loro dimensione "intrinseca", la tabella diventa scorrevole.

### 6.31.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando una nuova cella viene selezionata con i tasti.

#### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

### 6.31.5 Tasti

I seguenti *tasti* vengono elaborati dai Widget Table:

- `LV_KEY_RIGHT/LEFT/UP/DOWN` Seleziona una cella.

Si noti che, come di consueto, lo stato di `LV_KEY_ENTER` viene tradotto in `LV_EVENT_PRESSED/PRESSING/RELEASED` ecc.

`lv_table_get_selected_cell(table, &row, &col)` può essere utilizzato per ottenere la cella attualmente selezionata. Riga e colonna verranno impostate su `LV_TABLE_CELL_NONE` se non è selezionata alcuna cella.

#### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

### 6.31.6 Esempio

#### Tabella semplice

```
#include "../../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

static void draw_event_cb(lv_event_t * e)
{
    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_dsc(draw_task);
    /*If the cells are drawn...*/
    if(base_dsc->part == LV_PART_ITEMS) {
        uint32_t row = base_dsc->id1;
        uint32_t col = base_dsc->id2;

        /*Make the texts in the first cell center aligned*/
        if(row == 0) {
            lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
            if(label_draw_dsc) {
                label_draw_dsc->align = LV_TEXT_ALIGN_CENTER;
            }
            lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
            if(fill_draw_dsc) {
                fill_draw_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_BLUE), fill_draw_dsc->color, LV_OPA_20);
                fill_draw_dsc->opa = LV_OPA_COVER;
            }
        }
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

} /*In the first column align the texts to the right/
else if(col == 0) {
    lv_draw_label_dsc_t * label_draw_dsc = lv_draw_task_get_label_dsc(draw_task);
    if(label_draw_dsc) {
        label_draw_dsc->align = LV_TEXT_ALIGN_RIGHT;
    }
}

/*Make every 2nd row grayish/
if((row != 0 && row % 2) == 0) {
    lv_draw_fill_dsc_t * fill_draw_dsc = lv_draw_task_get_fill_dsc(draw_task);
    if(fill_draw_dsc) {
        fill_draw_dsc->color = lv_color_mix(lv_palette_main(LV_PALETTE_GREY), fill_draw_dsc->color, LV_OPA_10);
        fill_draw_dsc->opa = LV_OPA_COVER;
    }
}
}

void lv_example_table_1(void)
{
    lv_obj_t * table = lv_table_create(lv_screen_active());

    /*Fill the first column*/
    lv_table_set_cell_value(table, 0, 0, "Name");
    lv_table_set_cell_value(table, 1, 0, "Apple");
    lv_table_set_cell_value(table, 2, 0, "Banana");
    lv_table_set_cell_value(table, 3, 0, "Lemon");
    lv_table_set_cell_value(table, 4, 0, "Grape");
    lv_table_set_cell_value(table, 5, 0, "Melon");
    lv_table_set_cell_value(table, 6, 0, "Peach");
    lv_table_set_cell_value(table, 7, 0, "Nuts");

    /*Fill the second column*/
    lv_table_set_cell_value(table, 0, 1, "Price");
    lv_table_set_cell_value(table, 1, 1, "$7");
    lv_table_set_cell_value(table, 2, 1, "$4");
    lv_table_set_cell_value(table, 3, 1, "$6");
    lv_table_set_cell_value(table, 4, 1, "$2");
    lv_table_set_cell_value(table, 5, 1, "$5");
    lv_table_set_cell_value(table, 6, 1, "$1");
    lv_table_set_cell_value(table, 7, 1, "$9");

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_height(table, 200);
    lv_obj_center(table);

    /*Add an event callback to to apply some custom drawing*/
    lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_flag(table, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);
}

#endif

```

## Elenco leggero da tabella

```

#include "../../lv_examples.h"
#if LV_USE_TABLE && LV_BUILD_EXAMPLES

#define ITEM_CNT 200

static void draw_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);

    lv_draw_task_t * draw_task = lv_event_get_draw_task(e);
    lv_draw_dsc_base_t * base_dsc = (lv_draw_dsc_base_t *)lv_draw_task_get_draw_dsc(draw_task);
    /*If the cells are drawn...*/
    if(base_dsc->part == LV_PART_ITEMS && lv_draw_task_get_type(draw_task) == LV_DRAW_TASK_TYPE_FILL) {
        /*Draw the background*/
        bool chk = lv_table_has_cell_ctrl(obj, base_dsc->id1, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
        lv_draw_rect_dsc_t rect_dsc;
        lv_draw_rect_dsc_init(&rect_dsc);
        rect_dsc.bg_color = chk ? lv_theme_get_color_primary(obj) : lv_palette_lighten(LV_PALETTE_GREY, 2);
        rect_dsc.radius = LV_RADIUS_CIRCLE;

        lv_area_t sw_area;
        sw_area.x1 = 0;
        sw_area.x2 = 40;
        sw_area.y1 = 0;
        sw_area.y2 = 24;
        lv_area_t draw_task_area;
        lv_draw_task_get_area(draw_task, &draw_task_area);
        lv_area_align(&draw_task_area, &sw_area, LV_ALIGN_RIGHT_MID, -15, 0);
        lv_draw_rect(base_dsc->layer, &rect_dsc, &sw_area);

        /*Draw the knob*/
        rect_dsc.bg_color = lv_color_white();
        lv_area_t knob_area;
        knob_area.x1 = 0;
        knob_area.x2 = 18;
        knob_area.y1 = 0;
        knob_area.y2 = 18;
        if(chk) {
            lv_area_align(&sw_area, &knob_area, LV_ALIGN_RIGHT_MID, -3, 0);
        } else {
            lv_area_align(&sw_area, &knob_area, LV_ALIGN_LEFT_MID, 3, 0);
        }
        lv_draw_rect(base_dsc->layer, &rect_dsc, &knob_area);
    }
}

static void change_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    uint32_t col;

```

(continues on next page)

(continua dalla pagina precedente)

```

uint32_t row;
lv_table_get_selected_cell(obj, &row, &col);
bool chk = lv_table_has_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
if(chk) lv_table_clear_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
else lv_table_set_cell_ctrl(obj, row, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);
}

/**
 * A very light-weighted list created from table
 */
void lv_example_table_2(void)
{
    /*Measure memory usage*/
    lv_mem_monitor_t mon1;
    lv_mem_monitor(&mon1);

    uint32_t t = lv_tick_get();

    lv_obj_t * table = lv_table_create(lv_screen_active());

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_size(table, LV_SIZE_CONTENT, 200);

    lv_table_set_column_width(table, 0, 150);
    lv_table_set_row_count(table, ITEM_CNT); /*Not required but avoids a lot of memory reallocation lv_table_set_set_value*/
    lv_table_set_column_count(table, 1);

    /*Don't make the cell pressed, we will draw something different in the event*/
    lv_obj_remove_style(table, NULL, LV_PART_ITEMS | LV_STATE_PRESSED);

    uint32_t i;
    for(i = 0; i < ITEM_CNT; i++) {
        lv_table_set_cell_value_fmt(table, i, 0, "Item %" LV_PRIu32, i + 1);
    }

    lv_obj_align(table, LV_ALIGN_CENTER, 0, -20);

    /*Add an event callback to to apply some custom drawing*/
    lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_TASK_ADDED, NULL);
    lv_obj_add_event_cb(table, change_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_add_flag(table, LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS);

    lv_mem_monitor_t mon2;
    lv_mem_monitor(&mon2);

    size_t mem_used = mon1.free_size - mon2.free_size;

    uint32_t elaps = lv_tick_elaps(t);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text_fmt(label, "%" LV_PRIu32 " items were created in %" LV_PRIu32 " ms\n"
                           "using %zu bytes of memory",
                           (uint32_t)ITEM_CNT, elaps, mem_used);

    lv_obj_align(label, LV_ALIGN_BOTTOM_MID, 0, -10);
}
#endif

```

## MicroPython

Nessun esempio ancora.

### 6.31.7 API

*lv\_api\_map\_v8.h*

*lv\_table.h*

*lv\_table\_private.h*

## 6.32 Tab View (lv\_tabview)

### 6.32.1 Panoramica

Il Widget "Tab View" [Visualizzazione schede] può essere utilizzato per organizzare i contenuti in schede. La "Tab View" è composta da altri Widget:

- Contenitore principale: *Widget Base (lv\_obj)*
- Pulsanti Tab: un *Widget Base (lv\_obj)* con *Button (lv\_button)*
- Contenitore per le schede: *Widget Base (lv\_obj)*
- Contenuto delle schede: *Widget Base (lv\_obj)*

I pulsanti scheda possono essere posizionati in alto, in basso, a sinistra e a destra della "Tab View".

È possibile selezionare una nuova scheda cliccando su un pulsante scheda o scorrendo orizzontalmente sul contenuto.

## 6.32.2 Parti e Stili

Non ci sono parti specifiche nella "Tab View", ma i Widget *Widget Base (lv\_obj)* e *Button (lv\_button)* engono utilizzati per creare la "Tab View".

## 6.32.3 Utilizzo

### Creazione di una Tab View

`lv_tabview_create(parent)` crea una nuova Tab View vuota.

### Aggiunta di schede

È possibile aggiungere nuove schede con `lv_tabview_add_tab(tabview, "Tab name")`. Questo restituirà un puntatore a un *Widget Base (lv\_obj)* n cui è possibile creare il contenuto della scheda.

### Rinominare le schede

Una scheda può essere rinominata con `lv_tabview_rename_tab(tabview, tab_id, "New Name")`.

### Passare a una nuova scheda

Per selezionare una nuova scheda si può:

- Cliccare sul pulsante della scheda
- Scorrere orizzontalmente
- Usare la funzione `lv_tabview_set_active(tabview, tab_id, LV_ANIM_ON / OFF)`

### Impostare la posizione della barra delle schede

Utilizzando la funzione `lv_tabview_set_tab_bar_position(tabview, LV_DIR_LEFT / RIGHT / TOP / BOTTOM)` la barra delle schede può essere spostata su qualsiasi lato.

### Impostare le dimensioni della barra delle schede

Le dimensioni della barra delle schede possono essere regolate tramite `lv_tabview_set_tab_bar_size(tabview, size)`. Quando le schede sono in alto o in basso, questo si riferisce all'altezza della barra delle schede, mentre quando sono sui lati, si riferisce alla larghezza.

### Accesso alle parti

- `lv_tabview_get_content(tabview)` restituisce un puntatore al contenitore per il contenuto delle schede (un *Widget Base (lv\_obj)*)
- `lv_tabview_get_tab_bar(tabview)` restituisce un puntatore al contenitore per i pulsanti delle schede (un *Widget Base (lv\_obj)*)

## 6.32.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato quando si seleziona una nuova scheda facendo scorrere o cliccando sul pulsante della scheda. `lv_tabview_get_tab_active(tabview)` restituisce l'indice a base zero della scheda corrente.

### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

## 6.32.5 Tasti

I tasti hanno effetto solo sui pulsanti delle schede. Aggiungere questi pulsanti a un gruppo tramite codice, se necessario.

### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

## 6.32.6 Esempio

### Tabview Semplice

```
#include "../../lv_examples.h"
#if LV_USE_TABVIEW && LV_BUILD_EXAMPLES

void lv_example_tabview_1(void)
{
    /*Create a Tab view object*/
    lv_obj_t * tabview;
    tabview = lv_tabview_create(lv_screen_active());

    /*Add 3 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t * tab1 = lv_tabview_add_tab(tabview, "Tab 1");
    lv_obj_t * tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t * tab3 = lv_tabview_add_tab(tabview, "Tab 3");

    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "This the first tab\n\n"
                        "If the content\n"
                        "of a tab\n"
                        "becomes too\n"
                        "longer\n"
                        "than the\n"
                        "container\n"
                        "then it\n"
                        "automatically\n"
                        "becomes\n"
                        "scrollable.\n"
                        "\n"
                        "\n"
                        "\n"
                        "Can you see it?");

    label = lv_label_create(tab2);
    lv_label_set_text(label, "Second tab");

    label = lv_label_create(tab3);
    lv_label_set_text(label, "Third tab");

    lv_obj_scroll_to_view_recursive(label, LV_ANIM_ON);
}

#endif
```

### Tab a sinistra, con stile e nessuno scorrimento

```
#include "../../lv_examples.h"
#if LV_USE_TABVIEW && LV_BUILD_EXAMPLES

/*A vertical tab view with disabled scrolling and some styling*/
void lv_example_tabview_2(void)
{
    /*Create a Tab view object*/
    lv_obj_t * tabview;
    uint32_t tab_count = 0;
    uint32_t i = 0;

    tabview = lv_tabview_create(lv_screen_active());
    lv_tabview_set_tab_bar_position(tabview, LV_DIR_LEFT);
    lv_tabview_set_tab_bar_size(tabview, 80);

    lv_obj_set_style_bg_color(tabview, lv_palette_lighten(LV_PALETTE_RED, 2), 0);
    lv_obj_t * tab_buttons = lv_tabview_get_tab_bar(tabview);
    lv_obj_set_style_bg_color(tab_buttons, lv_palette_darken(LV_PALETTE_GREY, 3), 0);
    lv_obj_set_style_text_color(tab_buttons, lv_palette_lighten(LV_PALETTE_GREY, 5), 0);

    /*Add 5 tabs (the tabs are page (lv_page) and can be scrolled*/
    lv_obj_t * tab1 = lv_tabview_add_tab(tabview, "Tab 1");
    lv_obj_t * tab2 = lv_tabview_add_tab(tabview, "Tab 2");
    lv_obj_t * tab3 = lv_tabview_add_tab(tabview, "Tab 3");
    lv_obj_t * tab4 = lv_tabview_add_tab(tabview, "Tab 4");
    lv_obj_t * tab5 = lv_tabview_add_tab(tabview, "Tab 5");

    tab_count = lv_tabview_get_tab_count(tabview);
    for(i = 0; i < tab_count; i++) {
        lv_obj_t * button = lv_obj_get_child(tab_buttons, i);
        lv_obj_set_style_border_side(button, LV_BORDER_SIDE_RIGHT, LV_PART_MAIN | LV_STATE_CHECKED);
    }
    lv_obj_set_style_bg_color(tab2, lv_palette_lighten(LV_PALETTE_AMBER, 3), 0);
    lv_obj_set_style_bg_opa(tab2, LV_OPA_COVER, 0);

    /*Add content to the tabs*/
    lv_obj_t * label = lv_label_create(tab1);
    lv_label_set_text(label, "First tab");
```

(continues on next page)

(continua dalla pagina precedente)

```

label = lv_label_create(tab2);
lv_label_set_text(label, "Second tab");

label = lv_label_create(tab3);
lv_label_set_text(label, "Third tab");

label = lv_label_create(tab4);
lv_label_set_text(label, "Fourth tab");

label = lv_label_create(tab5);
lv_label_set_text(label, "Fifth tab");

lv_obj_remove_flag(lv_tabview_get_content(tabview), LV_OBJ_FLAG_SCROLLABLE);
#endif
}

```

## 6.32.7 API

*lv\_api\_map\_v8.h*

*lv\_tabview.h*

*lv\_tabview\_private.h*

## 6.33 Text Area (*lv\_textarea*)

### 6.33.1 Panoramica

La Text Area è un *Widget Base* (*lv\_obj*) con una *Label* (*lv\_label*) e un cursore. È possibile aggiungervi testo o caratteri. Le righe lunghe vengono andate a capo e, quando il testo diventa sufficientemente lungo, è possibile scorrere la "Text Area".

Sono supportate la modalità a riga singola e la modalità password.

### 6.33.2 Parti e Stili

- *LV\_PART\_MAIN* Il background della Text Area; utilizza le *tipiche proprietà di stile del background* e le proprietà di stile relative al testo, tra cui *text\_align*, per allineare il testo a sinistra, a destra o al centro.
- *LV\_PART\_SCROLLBAR* La scrollbar che viene visualizzata quando il testo è più lungo della sua altezza.
- *LV\_PART\_SELECTED* Determina lo stile del *testo selezionato*. È possibile utilizzare solo le proprietà di stile *text\_color* e *bg\_color*. *bg\_color* deve essere impostato direttamente sull'etichetta della Text Area.
- *LV\_PART\_CURSOR* Indica la posizione in cui vengono inseriti i caratteri. L'area del cursore è sempre il riquadro di delimitazione del carattere corrente. È possibile creare un cursore a blocco aggiungendo un colore di sfondo e un'opacità di sfondo allo stile di *LV\_PART\_CURSOR*. Per creare un cursore a "barra", lasciare il cursore trasparente e impostare un bordo sinistro. La proprietà di stile *anim\_time* imposta il tempo di lampeggio del cursore.
- *LV\_PART\_TEXTAREA\_PLACEHOLDER* Unica per la Text Area; consente di applicare uno stile al *testo segnaposto*.

### 6.33.3 Utilizzo

#### Aggiunta di testo

Si può inserire testo o caratteri nella posizione corrente del cursore con:

- *lv\_textarea\_add\_char*(*textarea*, 'c')
- *lv\_textarea\_add\_text*(*textarea*, "insert this text")

Per aggiungere caratteri larghi come 'á', 'ß' o caratteri CJK, si usa *lv\_textarea\_add\_text*(*textarea*, "á").

*lv\_textarea\_set\_text*(*textarea*, "New text") sostituisce tutto il testo esistente con "New text".

## Testo segnaposto

Il testo segnaposto è il testo che viene visualizzato quando la Text Area è vuota. Questo può essere un modo pratico per fornire all'utente finale un suggerimento su cosa digitare.

Specificare il testo segnaposto utilizzando `lv_textarea_set_placeholder_text(textarea, "Placeholder text")`.

## Eliminare un carattere

Per eliminare il carattere a sinistra della posizione corrente del cursore, utilizzare `lv_textarea_delete_char(textarea)`.

Per eliminare a destra, utilizzare `lv_textarea_delete_char_forward(textarea)`

## Spostamento del cursore

La posizione del cursore può essere modificata a livello di codice utilizzando `lv_textarea_set_cursor_pos(textarea, cursor_pos)` dove `cursor_pos` è l'indice a partire da zero del carattere davanti al quale deve essere posizionato il cursore. `LV_TEXTAREA_CURSOR_LAST` può essere passato per indicare "dopo l'ultimo carattere".

È possibile spostare il cursore di una posizione (o riga) alla volta con

- `lv_textarea_cursor_right(textarea)`
- `lv_textarea_cursor_left(textarea)`
- `lv_textarea_cursor_up(textarea)`
- `lv_textarea_cursor_down(textarea)`

Se si applica `lv_textarea_set_cursor_click_pos(textarea, true)`, il cursore salterà nella posizione in cui è stato cliccato sull'area di testo.

## Nascondere il cursore

Il cursore è normalmente sempre visibile. Può essere una buona idea impostarne lo stile in modo che sia visibile solo nello stato `LV_STATE_FOCUSED`. Vedere gli [Stili](#) per maggiori informazioni su come farlo.

## Modalità a una riga

La Text Area può essere configurata per mantenere tutto il testo su una singola riga con `lv_textarea_set_one_line(textarea, true)`. In questa modalità:

- l'altezza viene impostata automaticamente per visualizzare una sola riga,
- i caratteri di interruzione di riga vengono ignorati e
- il ritorno a capo automatico è disabilitato.

## Modalità password

La Text Area supporta la modalità password, che può essere abilitata con `lv_textarea_set_password_mode(textarea, true)`.

Per default, se il carattere • (Bullet, U+2022) è presente nel font, i caratteri immessi vengono convertiti in tale carattere dopo un ritardo configurabile dopo ogni nuovo carattere immesso. Se • non è presente nel font, verrà utilizzato \*. È possibile sovrascrivere il carattere di "mascheramento" di default con `lv_textarea_set_password_bullet(textarea, str)` dove `str` è una stringa C terminata da NUL. Esempio:

```
lv_textarea_set_password_bullet(textarea, "x");
```

In modalità password, `lv_textarea_get_text(textarea)` restituisce il testo effettivamente inserito, non i caratteri del punto elenco.

Il tempo di visibilità può essere regolato con `LV_TEXTAREA_DEF_PWD_SHOW_TIME` in `lv_conf.h`.

### Caratteri accettati

È possibile impostare un elenco di caratteri accettati con `lv_textarea_set_accepted_chars(textarea, list)` dove `list` è un puntatore a una stringa terminata da NULL, oppure NULL per accettare tutti i caratteri. I caratteri inseriti non presenti in questo elenco verranno ignorati.

```
lv_textarea_set_accepted_chars(textarea, "0123456789.+-");
```

### Lunghezza massima del testo

Il numero massimo di caratteri può essere limitato utilizzando `lv_textarea_set_max_length(textarea, max_char_num)`.

### Testo molto lungo

Se il testo nella Text Area è molto lungo (ad esempio > 20.000 caratteri), lo scorrimento e il disegno potrebbero risultare lenti. Tuttavia, impostando `LV_LABEL_LONG_TXT_HINT` in `lv_conf.h` su un valore diverso da zero, le prestazioni con il testo lungo migliorano significativamente. Questo risultato viene ottenuto salvando alcune informazioni aggiuntive sulla posizione verticale corrente del testo visualizzato. Con questa modalità configurata, lo scorrimento e il disegno sono veloci quanto con il testo breve "normale". Il costo è di 12 byte extra per etichetta in RAM.

Questo valore è impostato su 1 per default. Se non si utilizza testo lungo, è possibile risparmiare 12 byte per etichetta impostandolo su 0.

### Selezione del testo

Se `LV_LABEL_TEXT_SELECTION` è impostato su un valore diverso da zero in `lv_conf.h`, alcune funzionalità aggiuntive (e 8 byte per etichetta) vengono aggiunte ai Widget Label e ai Widget Text Area, e la funzionalità di selezione del testo viene automatizzata nei Widget Text Area. (Se non si utilizza testo selezionato nell'applicazione, è possibile risparmiare 8 byte per etichetta nella RAM impostando tale macro su 0.)

È possibile selezionare qualsiasi parte del testo se abilitata con `lv_textarea_set_text_selection(textarea, true)`. Funziona in modo molto simile a quando si seleziona del testo sul PC cliccando e trascinando con il mouse o un altro dispositivo di input puntatore. Se si passa `false` a questa funzione per disabilitare la selezione del testo, qualsiasi testo selezionato al momento della chiamata verrà deselectato.

In futuro si prevede di aggiungere la selezione del testo tramite Shift+click e la selezione del testo tramite tastiera con Shift+Freccia.

Se è necessario gestire il testo selezionato a livello di codice, oltre alla funzione `lv_textarea_set_text_selection(textarea, enable)`, ecco gli strumenti a disposizione. (`ta_label` è un puntatore all'etichetta della Text Area recuperata con `ta_label = lv_textarea_get_label(textarea);`)

- `lv_textarea_get_text_selection(textarea)` indica se la selezione del testo è abilitata.
- `lv_textarea_text_is_selected(textarea)` indica se è attualmente selezionato del testo.
- `lv_textarea_clear_selection(textarea)` cancella la selezione del testo corrente.
- `lv_label_set_text_selection_start(ta_label, index)` dove `index` è l'indice a partire da zero del primo carattere del testo selezionato. Passare `LV_DRAW_LABEL_NO_TXT_SEL` per specificare che non è stato selezionato alcun testo.
- `lv_label_set_text_selection_end(ta_label, index)` dove `index` è l'indice a partire da zero del carattere subito dopo il testo selezionato. Passare `LV_DRAW_LABEL_NO_TXT_SEL` per specificare che non è stato selezionato alcun testo.
- `lv_label_get_text_selection_start(ta_label)` ndice a partire da zero del primo carattere del testo selezionato. `LV_DRAW_LABEL_NO_TXT_SEL` indica che non è stato selezionato alcun testo.

- `lv_label_get_text_selection_end(ta_label)` indice a base zero del carattere subito dopo il testo selezionato. `LV_DRAW_LABEL_NO_TXT_SEL` indica che non è stato selezionato alcun testo.

Normalmente non ce n'è bisogno, poiché la Text Area automatizza il processo di selezione del testo, ma per modificare la selezione a livello di codice, gli strumenti sopra indicati sono quelli da usare.

### 6.33.4 Eventi

- `LV_EVENT_INSERT` Inviato subito prima dell'inserimento di un carattere o di un testo. Il parametro evento è il testo che sta per essere inserito. `lv_textarea_set_insert_replace(textarea, "New text")` può essere chiamato dall'interno di quell'evento per sostituire il testo da inserire. Il contenuto del buffer passato deve sopravvivere abbastanza a lungo da consentire il ritorno della chiamata a `lv_timer_handler()` che gestisce l'evento (dopodiché l'etichetta dell'area di testo avrà copiato il testo). Quindi non dovrebbe trattarsi di un buffer locale (creato nello stack) il cui contenuto verrà distrutto prima che ciò accada. Passare "" significa "non inserire nulla".
- `LV_EVENT_VALUE_CHANGED` Inviato quando il contenuto della Text Area è cambiato.
- `LV_EVENT_READY` Inviato quando `LV_KEY_ENTER` viene premuto (o inviato) a una Text Area di una sola riga.

#### Ulteriori Informazioni

Scoprire di più sugli [Eventi del Widget di Base](#) emessi da tutti i Widget.

Scoprire di più sugli [Eventi](#).

### 6.33.5 Tasti

- `LV_KEY_UP/DOWN/LEFT/RIGHT` Sposta il cursore
- Any character Aggiunge il carattere alla posizione corrente del cursore

#### Ulteriori Informazioni

Scoprire di più sui [Tasti](#).

### 6.33.6 Esempio

#### Area di testo semplice

```
#include "../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_BUILD_EXAMPLES

static void textarea_event_handler(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target_obj(e);
    LV_UNUSED(ta);
    LV_LOG_USER("Enter was pressed. The current text is: %s", lv_textarea_get_text(ta));
}

static void btm_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * ta = (lv_obj_t *)lv_event_get_user_data(e);
    const char * txt = lv_buttonmatrix_get_button_text(obj, lv_buttonmatrix_get_selected_button(obj));

    if(lv_strcmp(txt, LV_SYMBOL_BACKSPACE) == 0) lv_textarea_delete_char(ta);
    else if(lv_strcmp(txt, LV_SYMBOL_NEW_LINE) == 0) lv_obj_send_event(ta, LV_EVENT_READY, NULL);
    else lv_textarea_add_text(ta, txt);
}

void lv_example_textarea_1(void)
{
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_one_line(ta, true);
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, 10);
    lv_obj_add_event_cb(ta, textarea_event_handler, LV_EVENT_READY, ta);
    lv_obj_add_state(ta, LV_STATE_FOCUSED); /*To be sure the cursor is visible*/

    static const char * btm_map[] = {"1", "2", "3", "\n",
                                    "4", "5", "6", "\n",
                                    "7", "8", "9", "\n",
                                    LV_SYMBOL_BACKSPACE, "0", LV_SYMBOL_NEW_LINE, ""}
}
```

(continues on next page)

(continua dalla pagina precedente)

```

        };
        lv_obj_t * btm = lv_buttonmatrix_create(lv_screen_active());
        lv_obj_set_size(btm, 200, 150);
        lv_obj_align(btm, LV_ALIGN_BOTTOM_MID, 0, -10);
        lv_obj_add_event_cb(btm, btm_event_handler, LV_EVENT_VALUE_CHANGED, ta);
        lv_obj_remove_flag(btm, LV_OBJ_FLAG_CLICK_FOCUSABLE); /*To keep the text area focused on button clicks*/
        lv_buttonmatrix_set_map(btm, btm_map);
    }

#endif

```

## Area di testo con campo password

```

#include "../../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;

void lv_example_textarea_2(void)
{
    /*Create the password box*/
    lv_obj_t * pwd_ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_text(pwd_ta, "");
    lv_textarea_set_password_mode(pwd_ta, true);
    lv_textarea_set_one_line(pwd_ta, true);
    lv_obj_set_width(pwd_ta, lv_pct(40));
    lv_obj_set_pos(pwd_ta, 5, 20);
    lv_obj_add_event_cb(pwd_ta, ta_event_cb, LV_EVENT_ALL, NULL);

    /*Create a label and position it above the text box*/
    lv_obj_t * pwd_label = lv_label_create(lv_screen_active());
    lv_label_set_text(pwd_label, "Password:");
    lv_obj_align_to(pwd_label, pwd_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create the one-line mode text area*/
    lv_obj_t * text_ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_one_line(text_ta, true);
    lv_textarea_set_password_mode(text_ta, false);
    lv_obj_set_width(text_ta, lv_pct(40));
    lv_obj_add_event_cb(text_ta, ta_event_cb, LV_EVENT_ALL, NULL);
    lv_obj_align(text_ta, LV_ALIGN_TOP_RIGHT, -5, 20);

    /*Create a label and position it above the text box*/
    lv_obj_t * oneline_label = lv_label_create(lv_screen_active());
    lv_label_set_text(oneline_label, "Text:");
    lv_obj_align_to(oneline_label, text_ta, LV_ALIGN_OUT_TOP_LEFT, 0, 0);

    /*Create a keyboard*/
    kb = lv_keyboard_create(lv_screen_active());
    lv_obj_set_size(kb, LV_HOR_RES, LV_VER_RES / 2);

    lv_keyboard_set_textarea(kb, pwd_ta); /*Focus it on one of the text areas to start*/

    /*The keyboard will show Arabic characters if they are enabled */
    #if LV_USE_ARABIC_PERSIAN_CHARS && LV_FONT_DEJAVU_16_PERSIAN_HEBREW
        lv_obj_set_style_text_font(kb, &lv_font_dejavu_16_persian_hebrew, 0);
        lv_obj_set_style_text_font(text_ta, &lv_font_dejavu_16_persian_hebrew, 0);
        lv_obj_set_style_text_font(pwd_ta, &lv_font_dejavu_16_persian_hebrew, 0);
    #endif
}

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target_obj(e);
    if(code == LV_EVENT_CLICKED || code == LV_EVENT_FOCUSED) {
        /*Focus on the clicked text area*/
        if(kb != NULL) lv_keyboard_set_textarea(kb, ta);
    }

    else if(code == LV_EVENT_READY) {
        LV_LOG_USER("Ready, current text: %s", lv_textarea_get_text(ta));
    }
}
#endif

```

## Formattazione automatica del testo

```

#include "../../../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e);

static lv_obj_t * kb;

/***
 * Automatically format text like a clock. E.g. "12:34"
 * Add the ':' automatically.
 */
void lv_example_textarea_3(void)
{
    /*Create the text area*/
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_obj_add_event_cb(ta, ta_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_textarea_set_accepted_chars(ta, "0123456789:");
    lv_textarea_set_max_length(ta, 5);
    lv_textarea_set_one_line(ta, true);
    lv_textarea_set_text(ta, "");
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/*Create a keyboard*/
kb = lv_keyboard_create(lv_screen_active());
lv_obj_set_size(kb, LV_HOR_RES, LV_VER_RES / 2);
lv_keyboard_set_mode(kb, LV_KEYBOARD_MODE_NUMBER);
lv_keyboard_set_textarea(kb, ta);
}

static void ta_event_cb(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target_obj(e);
    const char * txt = lv_textarea_get_text(ta);
    if(txt[0] >= '0' && txt[0] <= '9' &&
       txt[1] >= '0' && txt[1] <= '9' &&
       txt[2] != ':') {
        lv_textarea_set_cursor_pos(ta, 2);
        lv_textarea_add_char(ta, ':');
    }
}

#endif

```

## Stile del cursore dell'area di testo

```

#include "../lv_examples.h"
#if LV_USE_TEXTAREA && LV_USE_KEYBOARD && LV_BUILD_EXAMPLES

static void create_styled_textarea_cursor(const char * txt, lv_coord_t y_ofs, lv_style_t * cursor_style)
{
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_textarea_set_text(ta, txt);
    lv_obj_set_width(ta, 280);
    lv_obj_align(ta, LV_ALIGN_TOP_MID, 0, y_ofs);
    lv_textarea_set_one_line(ta, true);
    lv_obj_add_state(ta, LV_STATE_FOCUSED);
    lv_obj_add_style(ta, cursor_style, LV_PART_CURSOR | LV_STATE_FOCUSED);
    lv_textarea_set_cursor_pos(ta, 0);
}

void lv_example_textarea_4(void)
{
    static lv_style_t style_simple, style_block, style_underline;

    /* Thin left bar cursor (simple) */
    lv_style_init(&style_simple);
    lv_style_set_border_color(&style_simple, lv_palette_main(LV_PALETTE_RED));

    /* Underline cursor */
    lv_style_init(&style_underline);
    lv_style_set_bg_opa(&style_underline, LV_OPA_TRANSP);
    lv_style_set_border_color(&style_underline, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_side(&style_underline, LV_BORDER_SIDE_BOTTOM);
    lv_style_set_pad_hor(&style_underline, 1); /* set width of cursor using pad */
    lv_style_set_border_width(&style_underline, 3); /* set thickness of underline cursor */

    /* Full block cursor with many styles */
    lv_style_init(&style_block);
    lv_style_set_bg_opa(&style_block, LV_OPA_COVER);
    lv_style_set_bg_color(&style_block, lv_palette_main(LV_PALETTE_ORANGE));
    lv_style_set_bg_grad_color(&style_block, lv_palette_main(LV_PALETTE_YELLOW));
    lv_style_set_bg_grad_dir(&style_block, LV_GRAD_DIR_VER);
    lv_style_set_border_color(&style_block, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_border_side(&style_block, LV_BORDER_SIDE_FULL);
    lv_style_set_border_width(&style_block, 1);
    lv_style_set_radius(&style_block, 4);
    lv_style_set_text_color(&style_block, lv_color_white());
    lv_style_set_pad_all(&style_block, 1); /* set width of cursor using pad */

    /* Create 3 independent textareas, each with a unique styled cursor */
    create_styled_textarea_cursor("This is a simple red cursor", 10, &style_simple);
    create_styled_textarea_cursor("This is an underline blue cursor", 110, &style_underline);
    create_styled_textarea_cursor("This is a complex block cursor", 60, &style_block);
}

#endif

```

## 6.33.7 API

*lv\_obj\_property\_names.h*

*lv\_textarea.h*

*lv\_textarea\_private.h*

## 6.34 Tile View (lv\_tileview)

### 6.34.1 Panoramica

La Tile View è un Widget contenitore i cui elementi (chiamati *tile*) possono essere disposti in una griglia. Un utente può navigare tra i tile scorrendo. Qualsiasi direzione di scorrimento può essere disabilitata sui tile singolarmente per impedire lo spostamento da un tile all'altro.

Se la Tile View è dimensionata per lo schermo, l'interfaccia utente assomiglia a quella che potreste aver visto sugli smartwatch.

### 6.34.2 Parti e Stili

La Tile View è composta da un contenitore *Widget Base (lv\_obj)* e da tile *Widget Base (lv\_obj)*.

Le parti e gli stili funzionano come per *Widget Base (lv\_obj)*.

### 6.34.3 Utilizzo

#### Aggiungere un tile

`lv_tileview_add_tile(tileview, col_id, row_id, dir)` crea un nuovo tile nella colonna `col_id`-esima e nella riga `row_id`-esima. `dir` può essere `LV_DIR_LEFT/RIGHT/TOP/BOTTOM/HOR/VER/ALL` o valori combinati OR per consentire lo spostamento ai tile [riquadri] adiacenti nella direzione specificata tramite scorrimento.

Il valore restituito è un puntatore al tile (un *Widget Base (lv\_obj)*) su cui è possibile creare il contenuto della scheda.

#### Combiomdi tile

La Tile View uò scorrere fino a un tile specificato con `lv_tileview_set_tile(tileview, tile, LV_ANIM_ON / OFF)` o con `lv_tileview_set_tile_by_index(tileview, col_id, row_id, LV_ANIM_ON / OFF)`.

### 6.34.4 Eventi

- `LV_EVENT_VALUE_CHANGED` Inviato dopo la visualizzazione di un nuovo tile tramite scorrimento. `lv_tileview_get_tile_active(tileview)` può essere utilizzato all'interno dell'evento per ottenere un puntatore al tile appena visualizzato.

#### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

### 6.34.5 Tasti

Nessun *Tasto* viene elaborato dai Widget Tileview.

#### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

### 6.34.6 Esempio

#### Tileview con contenuto

```
#include "../../lv_examples.h"
#if LV_USE_TILEVIEW && LV_BUILD_EXAMPLES

/** 
 * Create a 2x2 tile view and allow scrolling only in an "L" shape.
 * Demonstrate scroll chaining with a long list that
 * scrolls the tile view when it can't be scrolled further.
 */
void lv_example_tileview_1(void)
{
    lv_obj_t * tv = lv_tileview_create(lv_screen_active());

    /*Tile1: just a label*/
    lv_obj_t * tile1 = lv_tileview_add_tile(tv, 0, 0, LV_DIR_BOTTOM);
    lv_obj_t * label = lv_label_create(tile1);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_label_set_text(label, "Scroll down");
lv_obj_center(label);

/*Tile2: a button*/
lv_obj_t * tile2 = lv_tileview_add_tile(tv, 0, 1, (lv_dir_t)(LV_DIR_TOP | LV_DIR_RIGHT));
lv_obj_t * btn = lv_button_create(tile2);
label = lv_label_create(btn);
lv_label_set_text(label, "Scroll up or right");
lv_obj_set_size(btn, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
lv_obj_center(btn);

/*Tile3: a list*/
lv_obj_t * tile3 = lv_tileview_add_tile(tv, 1, 1, LV_DIR_LEFT);
lv_obj_t * list = lv_list_create(tile3);
lv_obj_set_size(list, LV_PCT(100), LV_PCT(100));

lv_list_add_button(list, NULL, "One");
lv_list_add_button(list, NULL, "Two");
lv_list_add_button(list, NULL, "Three");
lv_list_add_button(list, NULL, "Four");
lv_list_add_button(list, NULL, "Five");
lv_list_add_button(list, NULL, "Six");
lv_list_add_button(list, NULL, "Seven");
lv_list_add_button(list, NULL, "Eight");
lv_list_add_button(list, NULL, "Nine");
lv_list_add_button(list, NULL, "Ten");

}

#endif

```

## 6.34.7 API

*lv\_api\_map\_v8.h*  
*lv\_tileview.h*  
*lv\_tileview\_private.h*

## 6.35 Window (lv\_win)

### 6.35.1 Panoramica

Il Widget "Window" [finestra] è costituito da un'intestazione (simile a una barra del titolo) con titolo, pulsanti e un'area per i contenuti.

### 6.35.2 Parti e Stili

La Window è costruita a partire da altri Widget, quindi di può consultare la relativa documentazione per i dettagli:

- Background: *Widget Base (lv\_obj)*
- Intestazione sul background: *Widget Base (lv\_obj)*
- Titolo sull'intestazione: *Label (lv\_label)*
- Pulsanti sull'intestazione: *Button (lv\_button)*
- Content Area sul background: *Widget Base (lv\_obj)*

### 6.35.3 Utilizzo

#### Creazione di una Window

*lv\_win\_create(parent)* crea una Window inizialmente composta dalla seguente struttura widget:

- Background (un *Widget Base (lv\_obj)*, il contenitore della finestra principale), è configurato per essere un contenitore Flex-Flow che distribuisce verticalmente i Widget in esso contenuti (*LV\_FLEX\_FLOW\_COLUMN*).
- L'Header (Intestazione) (come una barra del titolo) è inizialmente vuota ed è un contenitore Flex-Flow configurato per disporre i Widget in essa contenuti orizzontalmente (*LV\_FLEX\_FLOW\_ROW*), da sinistra a destra. L'Intestazione occupa l'intera larghezza dello Background (il suo elemento padre) e la parte superiore per circa 1,25 cm (secondo *LV\_DPI\_DEF*).

- La "Content Area" (un *Widget Base* (*lv\_obj*)) occupa l'intera larghezza del Background (il suo elemento padre) e la parte rimanente dell'area del Background disponibile sotto l'Header. Non è *di per sé* un contenitore Flex-Flow, ma è possibile renderlo tale se lo si desidera. Vedere *Flex* per i dettagli.

## Titolo e pulsanti

È possibile aggiungere widget Button e Label all'Header utilizzando queste due funzioni. Verranno posizionati nell'intestazione in ordine da sinistra a destra man mano che vengono aggiunti. Queste funzioni possono essere chiamate in qualsiasi ordine e un numero qualsiasi di volte.

- *lv\_win\_add\_title*(*win*, "The title") aggiunge un'etichetta all'intestazione, con modalità estesa *LV\_LABEL\_LONG\_DOT* in modo che, se il contenuto del testo è più largo dell'area occupata, il testo verrà troncato con un'ellissi ("...") alla fine. È impostato su FLEX GROW 1, quindi se è l'unica Label nell'intestazione, occuperà tutto lo spazio disponibile dopo l'aggiunta di eventuali Pulsanti. Se vengono aggiunte più etichette, ciascuna condividerà equamente lo spazio, a meno che il suo valore FLEX GROW non venga modificato. *Flex* per i dettagli su come funziona. Per questo motivo, tutti i pulsanti aggiunti dopo l'ultima etichetta verranno "impilati" all'estrema destra dell'Header.
- *lv\_win\_add\_button*(*win*, *icon*, *button\_width*) aggiunge un Pulsante con la larghezza specificata che occupa l'intera altezza dell'Header (il suo elemento padre). Se *icon* non è NULL, viene creata un'immagine, centrata sul pulsante, utilizzando *icon* come sorgente. Sono supportate tutte le sorgenti di immagini valide, ma una sorgente comune da utilizzare è una delle macro *LV\_SYMBOL\_...*, come *LV\_SYMBOL\_CLOSE* per fornire un simbolo "x" (chiudi). Si ottiene un puntatore al Pulsante creato, in modo da poter aggiungere un callback di evento e/o qualsiasi altra cosa necessaria.

### 6.35.4 Ottener le parti

*lv\_win\_get\_header*(*win*) restituisce un puntatore all'intestazione, *lv\_win\_get\_content*(*win*) restituisce un puntatore al contenitore di contenuti a cui è possibile aggiungere il contenuto della finestra.

### 6.35.5 Eventi

I Widget Window non inviano eventi speciali, tuttavia è possibile aggiungere eventi a qualsiasi Pulsante aggiunto.

#### Ulteriori Informazioni

Scoprire di più sugli *Eventi del Widget di Base* emessi da tutti i Widget.

Scoprire di più sugli *Eventi*.

### 6.35.6 Tasti

Nessun *Tasto* viene elaborato dai Widget Window.

#### Ulteriori Informazioni

Scoprire di più sui *Tasti*.

### 6.35.7 Esempio

#### Window semplice

```
#include "../../lv_examples.h"
#if LV_USE_WIN && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    LV_UNUSED(obj);
    LV_LOG_USER("Button %d clicked", (int)lv_obj_get_index(obj));
}

```

(continues on next page)

(continua dalla pagina precedente)

```

void lv_example_win_1(void)
{
    lv_obj_t * win = lv_win_create(lv_screen_active());
    lv_obj_t * btn;
    btn = lv_win_add_button(win, LV_SYMBOL_LEFT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_win_add_title(win, "A title");

    btn = lv_win_add_button(win, LV_SYMBOL_RIGHT, 40);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    btn = lv_win_add_button(win, LV_SYMBOL_CLOSE, 60);
    lv_obj_add_event_cb(btn, event_handler, LV_EVENT_CLICKED, NULL);

    lv_obj_t * cont = lv_win_get_content(win); /*Content can be added here*/
    lv_obj_t * label = lv_label_create(cont);
    lv_label_set_text(label, "This is\n"
                      "a pretty\n"
                      "long text\n"
                      "to see how\n"
                      "the window\n"
                      "becomes\n"
                      "scrollable.\n"
                      "\n"
                      "\n"
                      "Some more\n"
                      "text to be\n"
                      "sure it\n"
                      "overflows. :)");
}

#endif

```

## 6.35.8 API

*lv\_windows\_context.h*  
*lv\_windows\_display.h*  
*lv\_windows\_input.h*  
*lv\_win.h*  
*lv\_win\_private.h*

## 6.36 Nuovo Widget

## Main Modules

---

### 7.1 Display (lv\_display)

#### 7.1.1 Panoramica

##### What is a Display?

In LVGL, an *lv\_display\_t* (not to be confused with a *Screen*) is a data type that represents a single display panel --- the hardware that displays LVGL-rendered pixels on your device.

##### How Many Displays Can LVGL Use?

LVGL can use any number of displays. It is only limited by available RAM and MCU time.

Why would you want multi-display support? Here are some examples:

- Have a "normal" TFT display with local UI and create "virtual" screens on VNC on demand. (You need to add your VNC driver.)
- Have a large TFT display and a small monochrome display.
- Have some smaller and simple displays in a large instrument or technology.
- Have two large TFT displays: one for a customer and one for the shop assistant.

If you set up LVGL to use more than one display, be aware that some functions use the *Display di Default* during their execution, such as creating *Screen*.

##### Attributi

Once created, a Display object remembers the characteristics of the display hardware it is representing, as well as other things relevant to its lifetime:

- Resolution (width and height in pixels)
- Color Depth (bits per pixel)
- Color Format (how colors in pixels are laid out)
- DPI (default is configured *LV\_DPI\_DEF* in *lv\_conf.h*, but can be modified with *lv\_display\_set\_dpi(disp, new\_dpi)*).
- 4 *Screen Layers* automatically created with each display

- All *Screen* created in association with this display (and not yet deleted---only one is displayed at any given time)
- The *Draw Buffer(s)* assigned to it
- The *Flush Callback* function that moves pixels from *Draw Buffer(s)* to Display hardware
- What areas of the display have been updated (made "dirty") so rendering logic can compute what to render during a *display refresh*
- Optional custom pointer as *User Data*

## User Data

With `lv_display_set_user_data(display1, p)` a custom pointer can be stored with `lv_display_t` object. This pointer can be used later, e.g. in *Eventi del Display*. See code example for how to do this in *Eventi del Display*.

## API

*lv\_display.h*

### 7.1.2 Setting Up Your Display(s)

During system initialization, you must do the following for each physical display panel you want LVGL to use:

- *create an lv\_display\_t* object for it,
- assign its *Draw Buffer(s)*, and
- assign a *Flush Callback* for it.

## Creating a Display

To create a display for LVGL:

```
lv_display_t * display1 = lv_display_create(hor_res, ver_res)
```

You can create *multiple displays* with a different driver for each (see below).

When an `lv_display_t` object is created, with it are created 4 Screens set up to help you manage layering of displayed Widgets. See *Transparent Screens* and *Screen Layers* for more information.

## Display di Default

Quando viene creato il primo oggetto *Display* (`lv_display`), questo diventa il Display di Default. If other Display Objects are created (to service additional Display Panels), the Default Display remains the first one created.

To set another *Display* (`lv_display`) as the Default Display, call `lv_display_set_default()`.

See *How Many Displays Can LVGL Use?* for more information about using multiple displays.

For many `lv_display_...` () functions, passing NULL for the `disp` argument will cause the function to target the Default Display. Check the API documentation for the function you are calling to be sure.

## Draw Buffer(s)

During system initialization, you must set drawing buffers for LVGL to use for each display. Do so by calling:

```
lv_display_set_buffers(display1, buf1, buf2, buf_size_in_bytes, render_mode)
```

- `buf1` a buffer to which LVGL can render pixels
- `buf2` a second optional buffer (see below)
- `buf_size_in_bytes` size of buffer(s) in bytes
- `render_mode` is one of the following:

- ***LV\_DISPLAY\_RENDER\_MODE\_PARTIAL*** Use the buffer(s) to render to the display using buffers smaller than the size of the display. Use of buffers at least 1/10 display size is recommended. In *Flush Callback* the rendered images need to be copied to the given area of the display. In this mode if a button is pressed only the button's area will be redrawn.
- ***LV\_DISPLAY\_RENDER\_MODE\_DIRECT*** The buffer size(s) must match the size of the display. LVGL will render into the correct location of the buffer. Using this method the buffer(s) always contain the whole display image. If two buffers are used, the rendered areas are automatically copied to the other buffer after flushing. Due to this in *Flush Callback* typically only a frame buffer address needs to be changed. If a button is pressed only the button's area will be redrawn.
- ***LV\_DISPLAY\_RENDER\_MODE\_FULL*** The buffer size(s) must match the size of the display. LVGL will always redraw the whole screen even if only 1 pixel has been changed. If two display-sized draw buffers are provided, LVGL's display handling works like "traditional" double buffering. This means the *Flush Callback* callback only has to update the address of the frame buffer to the `px_map` parameter.

## Simple Example

```
/* Declare buffer for 1/10 screen size; BYTES_PER_PIXEL will be 2 for RGB565. */
#define BYTES_PER_PIXEL (LV_COLOR_FORMAT_GET_SIZE(LV_COLOR_FORMAT_RGB565))
static uint8_t buf1[MY_DISP_HOR_RES * MY_DISP_VER_RES / 10 * BYTES_PER_PIXEL];
/* Set display buffer for display `display1` */
lv_display_set_buffers(display1, buf1, NULL, sizeof(buf1), LV_DISPLAY_RENDER_MODE_PARTIAL);
```

## One Buffer

If only one buffer is used, LVGL draws the content of the screen into that draw buffer and sends it to the display via the *Flush Callback*. LVGL then waits until `lv_display_flush_ready()` is called (that is, the content of the buffer has been sent to the display) before drawing something new into it.

## Two Buffers

If two buffers are used LVGL can draw into one buffer while the content of the other buffer is sent to the display in the background. DMA or other hardware should be used to transfer data to the display so the MCU can continue drawing. Doing so allows *rendering* and *refreshing* the display to become parallel operations.

## Three Buffers

Triple buffering enhances parallelism between rendering and data transfer compared to double buffering. When one buffer has completed rendering and another is actively undergoing DMA transfer, the third buffer enables immediate rendering of the next frame, eliminating CPU/GPU idle time caused by waiting for DMA completion. The third buffer is configured using the `lv_display_set_3rd_draw_buffer()` function.

## Flush Callback

Draw buffer(s) are simple array(s) that LVGL uses to render the display's content. Once rendering is has been completed, the content of the draw buffer is sent to the display using a Flush Callback function.

An example looks like this:

```
void my_flush_cb(lv_display_t * display, const lv_area_t * area, uint8_t * px_map)
{
    /* The most simple case (also the slowest) to send all rendered pixels to the
     * screen one-by-one. `put_px` is just an example. It needs to be implemented by you. */
    uint16_t * buf16 = (uint16_t *)px_map; /* Let's say it's a 16 bit (RGB565) display */
    int32_t x, y;
    for(y = area->y1; y <= area->y2; y++) {
        for(x = area->x1; x <= area->x2; x++) {
            put_px(x, y, *buf16);
            buf16++;
        }
    }

    /* IMPORTANT!!!
     * Inform LVGL that flushing is complete so buffer can be modified again. */
    lv_display_flush_ready(display);
}
```

During system initialization, tell LVGL you want that function to copy pixels from rendered pixel-buffers to a particular display by doing the following:

```
lv_display_set_flush_cb(display1, my_flush_cb)
```

Note that which display is targeted is passed to the function, so you can use the same function for multiple displays, or use different functions for multiple displays. It's up to you.

### Nota

`lv_display_flush_ready(display1)` needs to be called when flushing is complete to inform LVGL that the buffer is available again to render new content into it.

LVGL might render the screen in multiple chunks and therefore call your Flush Callback multiple times. To see whether the current call is for the last chunk being rendered, use `lv_display_flush_is_last(display1)`.

### Flush-Wait Callback

By using `lv_display_flush_ready()` LVGL will normally spin in a loop while waiting for flushing.

However with the help of `lv_display_set_flush_wait_cb()` a custom wait callback be set for flushing. This callback can use a semaphore, mutex, or anything else to optimize waiting for the flush to be completed. The callback need not call `lv_display_flush_ready()` since the caller takes care of that (clearing the display's **flushing** flag) when your callback returns.

However, if a Flush-Wait Callback is not set, LVGL assumes that `lv_display_flush_ready()` is called after the flush has completed.

### API

`lv_display.h`

`lv_display_private.h`

### 7.1.3 Screen Layers

When an `lv_display_t` object is created, 4 permanent *Screen* that facilitate layering are created and attached to it.

1. Bottom Layer (below Active Screen, transparent, not scroll-able, but click-able)
2. *Lo Screen Attivo*
3. Top Layer (above Active Screen, transparent and neither scroll-able nor click-able)
4. System Layer (above Top Layer, transparent and neither scroll-able nor click-able)

1, 3 and 4 are independent of the *Lo Screen Attivo* and they will be shown (if they contain anything that is visible) regardless of which screen is the *Lo Screen Attivo*.

### Nota

For the bottom layer to be visible, the Active Screen's background has to be at least partially, if not fully, transparent.

You can get pointers to each of these screens on the *Display di Default* by using (respectively):

- `lv_screen_active()`,
- `lv_layer_top()`,
- `lv_layer_sys()`, and
- `lv_layer_bottom()`.

You can get pointers to each of these screens on a specified display by using (respectively):

- `lv_display_get_screen_active(disp)`,
- `lv_display_get_layer_top(disp)`,
- `lv_display_get_layer_sys(disp)`, and
- `lv_display_get_layer_bottom(disp)`.

To set a Screen you create to be the *Lo Screen Attivo*, call `lv_screen_load()` or `lv_screen_load_anim()`.

## Top and System Layers

LVGL uses the Top Layer and System Layer to empower you to ensure that certain *Tutti i Widget* are *always* on top of other layers.

You can add "pop-up windows" to the *Top Layer* freely. The Top Layer was meant to be used to create Widgets that are visible on all Screens shown on a Display. But, the *System Layer* is intended for system-level things (e.g. mouse cursor will be placed there with `lv_indev_set_cursor()`).

These layers work like any other Widget, meaning they have styles, and any kind of Widgets can be created in them.

### Nota

While the Top Layer and System Layer are created by their owning *Display* (`lv_display`) as not scrollable and not clickable, these behaviors can be overridden the same as any other Widget by using `lv_obj_set_scrollbar_mode(scr1, LV_SCROLLBAR_MODE_xxx)` and `lv_obj_add_flag(scr1, LV_OBJ_FLAG_CLICKABLE)` respectively.

If the `LV_OBJ_FLAG_CLICKABLE` flag is set on the Top Layer, then it will absorb all user clicks and acts as a modal Widget.

```
lv_obj_add_flag(lv_layer_top(), LV_OBJ_FLAG_CLICKABLE);
```

## Layer Inferiore

Similar to the Top- and System Layers, the Bottom Layer is also the full size of the Display, but it is located below the *Lo Screen Attivo*. It's visible only if the Active Screen's background opacity is < 255.

### Ulteriori Informazioni

*Transparent Screens.*

## API

`lv_display.h`

### 7.1.4 Color Format

The default color format of the display is set according to `LV_COLOR_DEPTH` (see `lv_conf.h`)

- `LV_COLOR_DEPTH 32`: XRGB8888 (4 bytes/pixel)
- `LV_COLOR_DEPTH 24`: RGB888 (3 bytes/pixel)
- `LV_COLOR_DEPTH 16`: RGB565 (2 bytes/pixel)
- `LV_COLOR_DEPTH 8`: L8 (1 bytes/pixel)
- `LV_COLOR_DEPTH 1`: I1 (1 bit/pixel) Only support for horizontal mapped buffers. See *Monochrome Displays* for more details:

The `color_format` can be changed with `lv_display_set_color_depth(display, LV_COLOR_FORMAT_...)`. Besides the default value `LV_COLOR_FORMAT_ARGB8888`, other formats can be used as well.

It's very important that draw buffer(s) should be large enough for the selected color format.

### Swapping Endian-ness

In case of RGB565 color format it might be required to swap the 2 bytes because the SPI, I2C or 8 bit parallel port periphery sends them in the wrong order.

The ideal solution is to configure the hardware to handle the 16-bit data with a different byte order, however if this is not possible `lv_draw_sw_rgb565_swap(buf, buf_size_in_px)` can be called in the *Flush Callback* to swap the bytes.

If you wish you can also write your own function, or use assembly instructions for the fastest possible byte swapping.

Note that this is not about swapping the Red and Blue channel but converting

RRRRR GGG | GGG BBBBB

to

GGG BBBBB | RRRRR GGG.

### Monochrome Displays

LVGL supports rendering directly in a 1-bit format for monochrome displays. To enable it, set `LV_COLOR_DEPTH 1` or use `lv_display_set_color_format(display, LV_COLOR_FORMAT_I1)`.

The `LV_COLOR_FORMAT_I1` format assumes that bytes are mapped to rows (i.e., the bits of a byte are written next to each other). The order of bits is MSB first, which means:



are represented on the display as:



Ensure that the LCD controller is configured accordingly.

Internally, LVGL rounds the redrawn areas to byte boundaries. Therefore, updated areas will:

- start on an  $N \times 8$  coordinate, and
- end on an  $N \times 8 - 1$  coordinate.

When setting up the buffers for rendering (`lv_display_set_buffers()`), make the buffer 8 bytes larger. This is necessary because LVGL reserves 2 x 4 bytes in the buffer, as these are assumed to be used as a palette.

To skip the palette, include the following line in your *Flush Callback* function: `px_map += 8`.

As usual, monochrome displays support partial, full, and direct rendering modes as well. In full and direct modes, the buffer size should be large enough for the whole screen, meaning `(horizontal_resolution * vertical_resolution / 8) + 8` bytes. As LVGL can not handle fractional width make sure to round the horizontal resolution to 8 bits (for example 90 to 96).

The `lv_draw_sw_i1_convert_to_vtiled()` function is used to convert a draw buffer in I1 color format from a row-wise (htiled) to a column-wise (vtiled) buffer layout. This conversion is necessary for certain display controllers that require a different draw buffer mapping. The function assumes that the buffer width and height are rounded to a multiple of 8. The bit order of the resulting vtiled buffer can be specified using the `bit_order_lsb` parameter.

For more details, refer to the implementation in `lv_draw_sw_i1_convert_to_vtiled()` in `src/draw/sw/lv_draw_sw.c`.

To ensure that the redrawn areas start and end on byte boundaries, you can add a rounder callback to your display driver. This callback will round the width and height to the nearest multiple of 8.

Here is an example of how to implement and set a rounder callback:

```
static void my_roundr_cb(lv_event_t *e)
{
    lv_area_t *area = lv_event_get_param(e);

    /* Round the height to the nearest multiple of 8 */
    area->y1 = (area->y1 & ~0x7);
    area->y2 = (area->y2 | 0x7);
}

lv_display_add_event_cb(display, my_roundr_cb, LV_EVENT_INVALIDATE_AREA, display);
```

In this example, the `my_roundr_cb` function rounds the coordinates of the redrawn area to the nearest multiple of 8. The `x1` and `y1` coordinates are rounded down, while the `x2` and `y2` coordinates are rounded up. This ensures that the width and height of the redrawn area are always multiples of 8.

## Transparent Screens

Usually, the opacity of the Screen is `LV_OPA_COVER` to provide a solid background for its children. If this is not the case (opacity < 100%) the display's `bottom_layer` will be visible. If the bottom layer's opacity is also not `LV_OPA_COVER` LVGL will have no solid background to draw.

This configuration (transparent Screen) could be useful to create, for example, on-screen display (OSD) menus where a video is played on a different hardware layer of the display panel, and a menu is overlaid on a higher layer.

To properly render a UI on a transparent Screen the Display's color format needs to be set to one with an alpha channel (for example `LV_COLOR_FORMAT_ARGB8888`).

In summary, to enable transparent screens and displays for OSD menu-like UIs:

- Set the screen's `bg_opa` to transparent: `lv_obj_set_style_bg_opa(lv_screen_active(), LV_OPA_TRANS, LV_PART_MAIN)`
- Set the bottom layer's `bg_opa` to transparent: `lv_obj_set_style_bg_opa(lv_layer_bottom(), LV_OPA_TRANS, LV_PART_MAIN)`
- Set a color format with alpha channel. E.g. `lv_display_set_color_format(display, LV_COLOR_FORMAT_ARGB8888)`

## API

`lv_obj_style_gen.h`

`lv_display.h`

`lv_draw_sw_utils.h`

`lv_conf_internal.h`

### 7.1.5 Refreshing

#### Default Refresh Behavior

Normally the dirty (a.k.a invalid) areas are checked and redrawn in every `LV_DEF_REFR_PERIOD` milliseconds (set in `lv_conf.h`). This happens as a result of a refresh `Timer` (`lv_timer`) created that gets created when the display is created, and is executed at that interval.

#### Decoupling the Display Refresh Timer

However, in some cases you might need more control on when display refreshing happens, for example:

- to synchronize rendering with VSYNC or the TE signal;
- to time display refreshes immediately after a single screen update of all widgets that needed to have their display data updated (i.e. only updated once immediately before display refresh to reduce CPU overhead).

You can do this in the following way:

```
/* Delete original display refresh timer */
lv_display_delete_refr_timer(display);

/* Call this to refresh dirty (changed) areas of the display. */
lv_display_refr_timer(NULL);
```

If you have multiple displays call `lv_display_set_default(display1)` to select the display to refresh before `lv_display_refr_timer(NULL)`.

#### Nota

`lv_timer_handler()` and `lv_display_refr_timer()` must not run at the same time.

If the performance monitor is enabled, the value of `LV_DEF_REFR_PERIOD` needs to be set to match the refresh period of the display to ensure that the statistical results are correct.

#### Forcing a Refresh

Normally the invalidated areas (marked for redrawing) are rendered in `lv_timer_handler()` in every `LV_DEF_REFR_PERIOD` milliseconds. However, by using `lv_refr_now(display)` you can tell LVGL to redraw the invalid areas immediately. The refreshing will happen in `lv_refr_now()` which might take longer.

The parameter of `lv_refr_now()` is a pointer to the display to refresh. If `NULL` is passed, all displays that have active refresh timers will be refreshed.

#### API

`lv_refr.h`  
`lv_display.h`  
`lv_api_map_v9_1.h`  
`lv_conf_internal.h`  
`lv_timer.h`

#### 7.1.6 Eventi del Display

`lv_display_add_event_cb(disp, event_cb, LV_EVENT..., user_data)` adds an event handler to a display.

If you added `user_data` to the Display, you can retrieve it in an event like this:

```
lv_display_t * display;
my_type_t * my_user_data;
display = (lv_display_t *)lv_event_get_current_target(e);
my_user_data = lv_display_get_user_data(display);
```

The following events are sent for Display (`lv_display_t`) objects:

- `LV_EVENT_INVALIDATE_AREA`: Un'area viene invalidata (contrassegnata per il ridisegno). `lv_event_get_param(e)` restituisce un puntatore a un oggetto `lv_area_t` con le coordinate dell'area invalidata. L'area può essere modificata liberamente se necessario per adattarla a un requisito specifico del display. Di solito necessario con i display monocromatici per invalidare  $N \times 8$  righe o colonne in un unico passaggio.
- `LV_EVENT_RESOLUTION_CHANGED`: Inviato quando la risoluzione cambia a causa di `lv_display_set_resolution()` o di `lv_display_set_rotation()`.
- `LV_EVENT_COLOR_FORMAT_CHANGED`: Inviato come risultato di qualsiasi chiamata a `lv_display_set_color_format()`.
- `LV_EVENT_REFR_REQUEST`: Inviato quando si è verificato un evento che richiede un ridisegno.
- `LV_EVENT_REFR_START`: Inviato prima dell'inizio di un ciclo di aggiornamento. Inviato anche se non c'è nulla da ridisegnare.

- *LV\_EVENT\_REFR\_READY*: Inviato al completamento dell'aggiornamento (dopo il rendering e la chiamata a *Flush Callback*). Inviato anche se non si è verificato alcun ridisegno.
- *LV\_EVENT\_RENDER\_START*: Inviato appena prima dell'inizio del rendering.
- *LV\_EVENT\_RENDER\_READY*: Inviato dopo il completamento del rendering (prima di chiamare *Flush Callback*)
- *LV\_EVENT\_FLUSH\_START*: Inviato prima della chiamata a *Flush Callback*.
- *LV\_EVENT\_FLUSH\_FINISH*: Inviato dopo il ritorno della chiamata a *Flush Callback*.
- *LV\_EVENT\_FLUSH\_WAIT\_START*: Inviato all'inizio della chiamata interna a *wait\_for\_flushing()* -- si verifica indipendentemente dal fatto che si verifichi effettivamente un'attesa. La chiamata termina immediatamente se *disp->flushing == 0*.
- *LV\_EVENT\_FLUSH\_WAIT\_FINISH*: Inviato quando la chiamata a *wait\_for\_flushing()* sta per terminare, indipendentemente dal fatto che si sia verificata un'attesa effettiva.

**API***lv\_display.h*

### 7.1.7 Changing Resolution

To set the resolution of the display after creation use *lv\_display\_set\_resolution(display, hor\_res, ver\_res)*

It's not mandatory to use the whole display for LVGL, however in some cases the physical resolution is important. For example, the touchpad still sees the whole resolution, and the values need to be converted to the active LVGL display area. So the physical resolution and the offset of the active area can be set with *lv\_display\_set\_physical\_resolution(display, hor\_res, ver\_res)* and *lv\_display\_set\_offset(display, x, y)*

**API***lv\_display.h*

### 7.1.8 Inactivity Measurement

A user's inactivity time is measured and stored with each *lv\_display\_t* object. Every use of an *Input Device* (if *associated with the display*) counts as activity. To get time elapsed since the last activity, use *lv\_display\_get\_inactive\_time(display1)*. If NULL is passed, the lowest inactivity time among all displays will be returned (in this case NULL does *not* mean the *Display di Default*).

You can manually trigger an activity using *lv\_display\_trigger\_activity(display1)*. If *display1* is NULL, the *Display di Default* will be used (**not all displays**).

**i Ulteriori Informazioni**

- *lv\_port\_disp\_template.c* for a template for your own driver.
- *Drawing* to learn more about how rendering works in LVGL.

**API***lv\_display.h*

## 7.1.9 Rotation

LVGL supports rotation of the display in 90 degree increments.

The orientation of the display can be changed with `lv_display_set_rotation(disp, LV_DISPLAY_ROTATION_xxx)` where `xxx` is 0, 90, 180 or 270. This will swap the horizontal and vertical resolutions internally according to the set degree; however, it will not perform the actual rotation. When changing the rotation, the `LV_EVENT_SIZE_CHANGED` event is emitted to allow for hardware reconfiguration. If your display panel and/or its driver chip(s) do not support rotation, `lv_draw_sw_rotate()` can be used to rotate the buffer in the `Flush Callback` function.

`lv_display_rotate_area(display, &area)` rotates the rendered area according to the current rotation settings of the display.

Note that in `LV_DISPLAY_RENDER_MODE_DIRECT` the small changed areas are rendered directly in the frame buffer so they cannot be rotated later. Therefore in direct mode only the whole frame buffer can be rotated.

In the case of `LV_DISPLAY_RENDER_MODE_PARTIAL` the small rendered areas can be rotated on their own before flushing to the frame buffer.

`LV_DISPLAY_RENDER_MODE_FULL` can work with rotation if the buffer(s) being rendered to are different than the buffer(s) being rotated to in the flush callback and the buffers being rendered to do not have a stride requirement.

Below is an example for rotating when the rendering mode is `LV_DISPLAY_RENDER_MODE_PARTIAL` and the rotated image should be sent to a **display controller**.

```
/*Rotate a partially rendered area to another buffer and send it*/
void flush_cb(lv_display_t * disp, const lv_area_t * area, uint8_t * px_map)
{
    lv_display_rotation_t rotation = lv_display_get_rotation(disp);
    lv_area_t rotated_area;
    if(rotation != LV_DISPLAY_ROTATION_0) {
        lv_color_format_t cf = lv_display_get_color_format(disp);
        /*Calculate the position of the rotated area*/
        rotated_area = *area;
        lv_display_rotate_area(disp, &rotated_area);
        /*Calculate the source stride (bytes in a line) from the width of the area*/
        uint32_t src_stride = lv_draw_buf_width_to_stride(lv_area_get_width(area), cf);
        /*Calculate the stride of the destination (rotated) area too*/
        uint32_t dest_stride = lv_draw_buf_width_to_stride(lv_area_get_width(&rotated_area), cf);
        /*Have a buffer to store the rotated area and perform the rotation*/
        static uint8_t rotated_buf[500*1014];
        int32_t src_w = lv_area_get_width(area);
        int32_t src_h = lv_area_get_height(area);
        lv_draw_sw_rotate(px_map, rotated_buf, src_w, src_h, src_stride, dest_stride, rotation, cf);
        /*Use the rotated area and rotated buffer from now on*/
        area = &rotated_area;
        px_map = rotated_buf;
    }
    my_set_window(area->x1, area->y1, area->x2, area->y2);
    my_send_colors(px_map);
}
```

Below is an example for rotating when the rendering mode is `LV_DISPLAY_RENDER_MODE_PARTIAL` and the image can be rotated directly into a **frame buffer of the LCD peripheral**.

```
/*Rotate a partially rendered area to the frame buffer*/
void flush_cb(lv_display_t * disp, const lv_area_t * area, uint8_t * px_map)
{
    lv_color_format_t cf = lv_display_get_color_format(disp);
    uint32_t px_size = lv_color_format_get_size(cf);
    /*Calculate the position of the rotated area*/
    lv_area_t rotated_area = *area;
    lv_display_rotate_area(disp, &rotated_area);
    /*Calculate the properties of the source buffer*/
    int32_t src_w = lv_area_get_width(area);
    int32_t src_h = lv_area_get_height(area);
    uint32_t src_stride = lv_draw_buf_width_to_stride(src_w, cf);
    /*Calculate the properties of the frame buffer*/
    int32_t fb_stride = lv_draw_buf_width_to_stride(disp->hor_res, cf);
    uint8_t * fb_start = my_fb_address;
    fb_start += rotated_area.y1 * fb_stride + rotated_area.x1 * px_size;
    lv_display_rotation_t rotation = lv_display_get_rotation(disp);
    if(rotation == LV_DISPLAY_ROTATION_0) {
        int32_t y;
        for(y = area->y1; y <= area->y2; y++) {
            lv_memcpy(fb_start, px_map, src_stride);
            px_map += src_stride;
            fb_start += fb_stride;
        }
    } else {
        lv_draw_sw_rotate(px_map, fb_start, src_w, src_h, src_stride, fb_stride, rotation, cf);
    }
}
```

**API***lv\_display.h*

### 7.1.10 Constraints on Redrawn Area

Some display controllers have specific requirements for the window area where the rendered image can be sent (e.g.,  $x1$  must be even, and  $x2$  must be odd).

In the case of monochrome displays,  $x1$  must be  $N \times 8$ , and  $x2$  must be  $N \times 8 - 1$ . (If the display uses `LV_COLOR_FORMAT_I1`, LVGL automatically applies this rounding. See [Monochrome Displays](#).)

The size of the invalidated (redrawn) area can be controlled as follows:

```
void rounder_event_cb(lv_event_t * e)
{
    lv_area_t * a = lv_event_get_invalidate_area(e);

    a->x1 = a->x1 & (~0x1); /* Ensure x1 is even */
    a->x2 = a->x2 | 0x1;   /* Ensure x2 is odd */
}

...
lv_display_add_event_cb(disp, rounder_event_cb, LV_EVENT_INVALIDATE_AREA, NULL);
```

**API***lv\_display.h*

### 7.1.11 Tiled Rendering

When multiple CPU cores are available and a large area needs to be redrawn, LVGL must identify independent areas that can be rendered in parallel.

For example, if there are 4 CPU cores, one core can draw the screen's background while the other 3 must wait until it is finished. If there are 2 buttons on the screen, those 2 buttons can be rendered in parallel, but 2 cores will still remain idle.

Due to dependencies among different areas, CPU cores cannot always be fully utilized.

To address this, LVGL can divide large areas that need to be updated into smaller tiles. These tiles are independent, making it easier to find areas that can be rendered concurrently.

Specifically, if there are 4 tiles and 4 cores, there will always be an independent area for each core within one of the tiles.

The maximum number of tiles can be set using the function `lv_display_set_tile_cnt(disp, cnt)`. The default value is `LV_DRAW_SW_DRAW_UNIT_CNT` (or 1 if software rendering is not enabled).

Small areas are not further divided into smaller tiles because the overhead of spinning up 4 cores would outweigh the benefits.

The ideal tile size is calculated as `ideal_tile_size = draw_buf_size / tile_cnt`. For example, in `LV_DISPLAY_RENDER_MODE_DIRECT` mode on an 800x480 screen, the display buffer is  $800 \times 480 = 375\text{k}$  pixels. If there are 4 tiles, the ideal tile size is approximately 93k pixels. Based on this, core utilization is as follows:

- 30k pixels: 1 core
- 90k pixels: 1 core
- 95k pixels: 2 cores (above 93k pixels, 2 cores are used)
- 150k pixels: 2 cores
- 200k pixels: 3 cores (above 186k pixels, 3 cores are used)
- 300k pixels: 4 cores (above 279k pixels, 4 cores are used)
- 375k pixels: 4 cores

In `LV_DISPLAY_RENDER_MODE_DIRECT`, the screen-sized draw buffer is divided by the tile count to determine the ideal tile sizes. If smaller areas are refreshed, it may result in fewer cores being used.

In `LV_DISPLAY_RENDER_MODE_FULL`, the maximum number of tiles is always created when the entire screen is refreshed.

In `LV_DISPLAY_RENDER_MODE_PARTIAL`, the partial buffer is divided into tiles. For example, if the draw buffer is 1/10th the size of the screen and there are 2 tiles, then 1/20th + 1/20th of the screen area will be rendered at once.

Tiled rendering only affects the rendering process, and the *Flush Callback* is called once for each invalidated area. Therefore, tiling is not visible from the flushing point of view.

## API

`lv_display.h`

### 7.1.12 Extending/Combining Displays

#### Mirroring a Display

To mirror the image of a display to another display, you don't need to use multi-display support. Just transfer the buffer received in the first display's *Flush Callback* to the other display as well.

#### Splitting an Image

You can create a larger virtual display from an array of smaller ones. You can create it by:

1. setting the resolution of the displays to the large display's resolution;
2. in *Flush Callback*, truncate and modify the `area` parameter for each display; and
3. send the buffer's content to each real display with the truncated area.

### 7.1.13 API

`lv_display.h`

## 7.2 Input Device (lv\_indev)

### 7.2.1 Creating an Input Device

To create an input device on the *Display di Default*:

```
/* Create and set up at least one display before you register any input devices. */
lv_indev_t * indev = lv_indev_create();          /* Create input device connected to Default Display. */
lv_indev_set_type(indev, LV_INDEV_TYPE...);        /* Touch pad is a pointer-like device. */
lv_indev_set_read_cb(indev, my_input_read);        /* Set driver function. */
```

If you have multiple displays, you will need to ensure the Default Display is set to the display your input device is "connected to" before making the above calls.

The `type` member can be:

- `LV_INDEV_TYPE_POINTER`: touchpad or mouse
- `LV_INDEV_TYPE_KEYPAD`: keyboard or keypad
- `LV_INDEV_TYPE_ENCODER`: encoder with left/right turn and push options
- `LV_INDEV_TYPE_BUTTON`: external buttons virtually pressing the screen

`my_input_read` is a function pointer which will be called periodically to report the current state of an input device to LVGL.

## Touchpad, Touch-Screen, Mouse or Any Pointer

Input devices that can click points on the display belong to the **POINTER** category. Here is an example of a simple input-device Read Callback function:

```
lv_indev_set_type(indev, LV_INDEV_TYPE_POINTER);
...
void my_input_read(lv_indev_t * indev, lv_indev_data_t * data)
{
    if(touchpad_pressed) {
        data->point.x = touchpad_x;
        data->point.y = touchpad_y;
        data->state = LV_INDEV_STATE_PRESSED;
    } else {
        data->state = LV_INDEV_STATE_RELEASED;
    }
}
```

## Mouse Cursor

Pointer input devices (like a mouse) can have a cursor.

```
...  
lv_indev_t * mouse_indev = lv_indev_create();  
...  
LV_IMAGE_DECLARE(mouse_cursor_icon); /* Declare the image source. */  
lv_obj_t * cursor_obj = lv_image_create(lv_screen_active()); /* Create image Widget for cursor. */  
lv_image_set_src(cursor_obj, &mouse_cursor_icon); /* Set image source. */  
lv_indev_set_cursor(mouse_indev, cursor_obj); /* Connect image to Input Device. */
```

Note that the cursor object should have `lv_obj_remove_flag(cursor_obj, LV_OBJ_FLAG_CLICKABLE)`. For images, *clicking* is disabled by default.

## Gestures

Pointer input devices can detect basic gestures. By default, most Widgets send gestures to their parents so they can be detected on the Screen Widget in the form of an `LV_EVENT_GESTURE` event. Ad esempio:

```
void my_event(lv_event_t * e)
{
    lv_obj_t * screen = lv_event_get_current_target(e);
    lv_dir_t dir = lv_indev_get_gesture_dir(lv_indev_active());
    switch(dir) {
        case LV_DIR_LEFT:
            ...
            break;
        case LV_DIR_RIGHT:
            ...
            break;
        case LV_DIR_TOP:
            ...
            break;
        case LV_DIR_BOTTOM:
            ...
            break;
    }
    ...
    lv_obj_add_event_cb(screen, my_event, LV_EVENT_GESTURE, NULL);
}
```

To prevent passing the gesture event to the parent from a Widget, use `lv_obj_remove_flag(widget, LV_OBJ_FLAG_GESTURE_BUBBLE)`.

Note that, gestures are not triggered if a Widget is being scrolled.

If you did some action on a gesture you can call `lv_indev_wait_release(lv_indev_active())` in the event handler to prevent LVGL sending further input-device-related events.

## Crown Behavior

A "Crown" is a rotary device typically found on smart watches.

When the user clicks somewhere and after that turns the rotary the last clicked widget will be either scrolled or its value will be incremented/decremented (e.g. in case of a slider).

As this behavior is tightly related to the last clicked widget, the crown support is an extension of the pointer input device. Just set `data->diff` to the number of turned steps and LVGL will automatically send the `LV_EVENT_ROTARY` event or scroll the widget based on the `editable` flag in the widget's class. Non-editable widgets are scrolled and for editable widgets the event is sent.

To get the steps in an event callback use `int32_t diff = lv_event_get_rotary_diff(e)`

The rotary sensitivity can be adjusted on 2 levels:

1. in the input device by the `indev->rotary_sensitivity` element (1/256 unit), and
2. by the `rotary_sensitivity` style property in the widget (1/256 unit).

The final diff is calculated like this:

```
diff_final = diff_in * (indev_sensitivity / 256) + (widget_sensitivity / 256);
```

For example, if both the indev and widget sensitivity is set to 128 (0.5), the input diff will be multiplied by 0.25. The value of the Widget will be incremented by that value or the Widget will be scrolled that amount of pixels.

## Multi-touch gestures

LVGL has the ability to recognize multi-touch gestures, when a gesture is detected a `LV_EVENT_GESTURE` is passed to the object on which the gesture occurred. Currently, these multi-touch gestures are supported:

- Two fingers pinch (up and down)
- Two fingers rotation
- Two fingers swipe (infinite)

To enable the multi-touch gesture recognition set the `LV_USE_GESTURE_RECOGNITION` option in the `lv_conf.h` file.

Currently, the system sends the events if the gestures are in one of the following states:

- `LV_INDEV_GESTURE_STATE_RECOGNIZED`: The gesture has been recognized and is now active.
- `LV_INDEV_GESTURE_STATE_ENDED`: The gesture has ended.

## Multi-touch gestures overview

To recognize multi touch gestures, recognizers are used. The structure `lv_indev_t` contains an array of recognizers, one per gesture type. These recognizers are initialized internally by `lv_indev_create` by calling `lv_indev_gesture_init_recognizers` after the indev device is created. The recognizers can then be configured to modify the gestures thresholds. These thresholds are used to be able to recognize the gesture only after the threshold have been reached. They can be set-up like this:

- `lv_indev_set_pinch_up_threshold(lv_indev_t * indev, float threshold)`: Set the pinch up (zoom in) threshold in pixels.
- `lv_indev_set_pinch_down_threshold(lv_indev_t * indev, float threshold)`: Set the pinch down (zoom out) threshold in pixels.
- `lv_indev_set_rotation_rad_threshold(lv_indev_t * indev, float threshold)`: Set the rotation angle threshold in radians.

The recognizers can then be updated to recognize the gestures by calling `lv_indev_gesture_recognizers_update`. This must be done in the user defined indev `read_cb`. This will iterate over the recognizers and stop once it detects a recognized or ended gesture. For now only one multi-touch gesture can be recognized/ended at a time.

Once the recognizers are updated, calling `lv_indev_gesture_recognizers_set_data` will update the `lv_indev_data_t` structure. It is meant to be done in the indev `read_cb`. This allows the future `lv_event_t` to be filled with multi-touch gesture info.

Here is an example of the `read_cb`:

```
/* The recognizer keeps the state of the gesture */
static lv_indev_gesture_recognizer_t recognizer;

/* An array that stores the collected touch events */
static lv_indev_touch_data_t touches[10];
```

(continues on next page)

(continua dalla pagina precedente)

```
/* A counter that needs to be incremented each time a touch event is received */
static uint8_t touch_cnt;

static void touch_read_callback(lv_indev_t * drv, lv_indev_data_t * data)
{
    lv_indev_touch_data_t * touch;

    lv_indev_update_recognizers(drv, &touches[0], touch_cnt);

    touch_cnt = 0;

    /* Set the gesture information, before returning to LVGL */
    lv_indev_gesture_recognizers_set_data(drv, data);
}
```

The user is in charge of collecting the necessary touches events from the driver until the indev `read_cb` is called. It must then convert the specific driver input to `lv_indev_touch_data_t` to be processed by the `read_cb` at a later point. Here is an example using `libinput`:

```
/** 
 * @brief Convert the libinput to lvgl's representation of touch event
 * @param ev a pointer to the lib input event
 */
static void touch_event_queue_add(struct libinput_event *ev)
{
    struct libinput_event_touch *touch_ev;
    lv_indev_touch_data_t *cur;
    lv_indev_touch_data_t *t;
    uint32_t time;
    int i;
    int id;
    int type;

    type = libinput_event_get_type(ev);
    touch_ev = libinput_event_get_touch_event(ev);
    id = libinput_event_touch_get_slot(touch_ev);
    time = libinput_event_touch_get_time(touch_ev);

    /* Get the last event for contact point */
    t = &touches[0];
    cur = NULL;

    for (i = 0; i < touch_cnt; i++) {
        if (t->id == id) {
            cur = t;
        }
        t++;
    }

    if (cur != NULL && cur->timestamp == time) {
        /* Previous event has the same timestamp - ignore duplicate event */
        return;
    }

    if (cur == NULL ||
        type == LIBINPUT_EVENT_TOUCH_UP ||
        type == LIBINPUT_EVENT_TOUCH_DOWN) {

        /* create new event */
        cur = &touches[touch_cnt];
        touch_cnt++;
    }

    switch (type) {
        case LIBINPUT_EVENT_TOUCH_DOWN:
        case LIBINPUT_EVENT_TOUCH_MOTION:
            cur->point.x = (int) libinput_event_touch_get_x_transformed(touch_ev, SCREEN_WIDTH);
            cur->point.y = (int) libinput_event_touch_get_y_transformed(touch_ev, SCREEN_HEIGHT);
            cur->state = LV_INDEV_STATE_PRESSED;
            break;

        case LIBINPUT_EVENT_TOUCH_UP:
            cur->state = LV_INDEV_STATE_RELEASED;
            cur->point.x = 0;
            cur->point.y = 0;
            break;
    }

    cur->timestamp = time;
    cur->id = id;
}

/** 
 * @brief Filter out libinput events that are not related to touches
 * @param ev a pointer to the lib input event
 */
static void process_libinput_event(struct libinput_event *ev)
{
    int type;

    type = libinput_event_get_type(ev);

    switch (type) {
        case LIBINPUT_EVENT_TOUCH_MOTION:
        case LIBINPUT_EVENT_TOUCH_DOWN:
        case LIBINPUT_EVENT_TOUCH_UP:
            /* Filter only touch events */
            touch_event_queue_add(ev);
            break;
        default:
            /* Skip an unrelated libinput event */
            return;
    }
}
```

From this setup, the user can now register events callbacks to react to `LV_EVENT_GESTURE`.

### Nota

A touch event is represented by the `lv_indev_touch_data_t` structure, the fields being 1:1 compatible with events emitted by the `libinput` library

## Handling multi-touch gesture events

Once a gesture is recognized or ended, a `LV_EVENT_GESTURE` is sent. The user can use these functions to gather more information about the gesture:

- `lv_event_get_gesture_type(lv_event_t * gesture_event)`: Get the type of the gesture. To be used to check which multi-touch gesture is currently reported.
- `lv_indev_gesture_state_t lv_event_get_gesture_state(lv_event_t * gesture_event, lv_indev_gesture_type_t type)`: Get the state of the gesture. It can be one of those:
  - `LV_INDEV_GESTURE_STATE_NONE`: The gesture is not active.
  - `LV_INDEV_GESTURE_STATE_RECOGNIZED`: The gesture is recognized and can be used.
  - `LV_INDEV_GESTURE_STATE_ENDED`: The gesture ended.

These functions allow the user to confirm the gesture is the expected one and that it is in a usable state. The user can then request the gestures values with the following functions:

- `lv_event_get_pinch_scale(lv_event_t * gesture_event)`: Get the pinch scale. Only relevant for pinch gesture.
- `lv_event_get_rotation(lv_event_t * gesture_event)`: Get the rotation in radians. Only relevant for rotation gesture.
- `lv_event_get_two_fingersSwipeDistance(lv_event_t * gesture_event)`: Get the distance in pixels from the gesture starting center. Only relevant for two fingers swipe gesture.
- `lv_event_get_two_fingersSwipeDir(lv_event_t * gesture_event)`: Get the direction from the starting center. Only relevant for two fingers swipe gesture.

This allows the user to react to the gestures and to use the gestures values. An example of such an application is available in the source tree `examples/others/gestures/lv_example_gestures.c`.

## Keypad or Keyboard

Full keyboards with all the letters or simple keypads with a few navigation buttons belong in the keypad category.

You can fully control the user interface without a touchpad or mouse by using a keypad or encoder. It works similar to the `TAB` key on the PC to select an element in an application or web page.

To use a keyboard or keypad:

- Register a Read Callback function for your device and set its type to `LV_INDEV_TYPE_KEYPAD`.
- Create a Widget Group (`lv_group_t * g = lv_group_create()`) and add Widgets to it with `lv_group_add_obj(g, widget)`.
- Assign the group to an input device: `lv_indev_set_group(indev, g)`.
- Use `LV_KEY_...` to navigate among the Widgets in the group. See `lv_core/lv_group.h` for the available keys.

```
lv_indev_set_type(indev, LV_INDEV_TYPE_KEYPAD);
...
void keyboard_read(lv_indev_t * indev, lv_indev_data_t * data){
    data->key = last_key(); /* Get the last pressed or released key */
```

(continues on next page)

(continua dalla pagina precedente)

```
if(key_pressed()) data->state = LV_INDEV_STATE_PRESSED;
else data->state = LV_INDEV_STATE_RELEASED;
```

## Encoder

A common example of an encoder is a device with a turning knob that tells the hosting device *when* the knob is being turned, and *in which direction*.

With an encoder your application can receive events from the following:

1. press of its button,
2. long-press of its button,
3. turn left, and
4. turn right.

In short, the Encoder input devices work like this:

- By turning the encoder you can focus on the next/previous object.
- When you press the encoder on a simple object (like a button), it will be clicked.
- If you press the encoder on a complex object (like a list, message box, etc.) the Widget will go to edit mode whereby you can navigate inside the object by turning the encoder.
- To leave edit mode, long press the button.

To use an Encoder (similar to the *Keypads*) the Widgets should be added to a group.

```
lv_indev_set_type(indev, LV_INDEV_TYPE_ENCODER);
...
void encoder_read(lv_indev_t * indev, lv_indev_data_t * data){
    data->enc_diff = enc_get_new_moves();
    if(enc_pressed()) data->state = LV_INDEV_STATE_PRESSED;
    else data->state = LV_INDEV_STATE_RELEASED;
}
```

## Widget Groups

When input focus needs to be managed among a set of Widgets (e.g. to capture user input from a keypad or encoder), that set of Widgets is placed in a group which thereafter manages how input focus moves from Widget to Widget.

In each group there is exactly one object with focus which receives the pressed keys or the encoder actions. For example, if a *Text Area* has focus and you press some letter on a keyboard, the keys will be sent and inserted into the text area. Similarly, if a *Slider* has focus and you press the left or right arrows, the slider's value will be changed.

You need to associate an input device with a group. An input device can send key events to only one group but a group can receive data from more than one input device.

To create a group use `lv_group_t * g = lv_group_create()` and to add a Widget to the group use `lv_group_add_obj(g, widget)`.

Once a Widget has been added to a group, you can find out what group it is in using `lv_obj_get_group(widget)`.

To find out what Widget in a group has focus, if any, call `lv_group_get_focused(group)`. If a Widget in that group has focus, it will return a pointer to it, otherwise it will return NULL.

To associate a group with an input device use `lv_indev_set_group(indev, g)`.

## Tasti

There are some predefined keys which have special meaning:

- `LV_KEY_NEXT`: Move focus to next object
- `LV_KEY_PREV`: Move focus to previous object
- `LV_KEY_ENTER`: Triggers `LV_EVENT_PRESSED`, `LV_EVENT_CLICKED`, or `LV_EVENT_LONG_PRESSED` etc. events
- `LV_KEY_UP`: Increase value or move up
- `LV_KEY_DOWN`: Decrease value or move down
- `LV_KEY_RIGHT`: Increase value or move to the right
- `LV_KEY_LEFT`: Decrease value or move to the left
- `LV_KEY_ESC`: Close or exit (e.g. close a *Drop-Down List*)
- `LV_KEY_DEL`: Delete (e.g. a character on the right in a *Text Area*)
- `LV_KEY_BACKSPACE`: Delete (e.g. a character on the left in a *Text Area*)
- `LV_KEY_HOME`: Go to the beginning/top (e.g. in a *Text Area*)
- `LV_KEY_END`: Go to the end (e.g. in a *Text Area*)

The most important special keys in your `read_cb()` function are:

- `LV_KEY_NEXT`
- `LV_KEY_PREV`
- `LV_KEY_ENTER`
- `LV_KEY_UP`
- `LV_KEY_DOWN`
- `LV_KEY_LEFT`
- `LV_KEY_RIGHT`

You should translate some of your keys to these special keys to support navigation in a group and interact with selected Widgets.

Usually, it's enough to use only `LV_KEY_LEFT` and `LV_KEY_RIGHT` because most Widgets can be fully controlled with them.

With an encoder you should use only `LV_KEY_LEFT`, `LV_KEY_RIGHT`, and `LV_KEY_ENTER`.

## Edit and Navigate Mode

Since a keypad has plenty of keys, it's easy to navigate between Widgets and edit them using the keypad. But encoders have a limited number of "keys" and hence it is difficult to navigate using the default options. *Navigate* and *Edit* modes are used to avoid this problem with encoders.

In *Navigate* mode, an encoder's `LV_KEY_LEFT` or `LV_KEY_RIGHT` is translated to `LV_KEY_NEXT` or `LV_KEY_PREV`. Therefore, the next or previous object will be selected by turning the encoder. Pressing `LV_KEY_ENTER` will change to *Edit* mode.

In *Edit* mode, `LV_KEY_NEXT` and `LV_KEY_PREV` is usually used to modify an object. Depending on the Widget's type, a short or long press of `LV_KEY_ENTER` changes back to *Navigate* mode. Usually, a Widget which cannot be pressed (like a *Slider*) leaves *Edit* mode upon a short click. But with Widgets where a short click has meaning (e.g. *Button*), a long press is required.

## Default Group

Interactive widgets (such as Buttons, Checkboxes, Sliders, etc.) can be automatically added to a default group. Just create a group with `lv_group_t * g = lv_group_create()` and set the default group with `lv_group_set_default(g)`

Don't forget to assign one or more input devices to the default group with `lv_indev_set_group(my_indev, g)`.

## Stili

When a Widget receives focus either by clicking it via touchpad or by navigating to it with an encoder or keypad, it goes to the `LV_STATE_FOCUSED` state. Hence, focused styles will be applied to it.

If a Widget switches to edit mode it enters the `LV_STATE_FOCUSED | LV_STATE_EDITED` states so any style properties assigned to these states will be shown.

See [Stili](#) for more details.

## Using Buttons with Encoder Logic

In addition to standard encoder behavior, you can also utilize its logic to navigate(focus) and edit widgets using buttons. This is especially handy if you have only few buttons available, or you want to use other buttons in addition to an encoder wheel.

You need to have 3 buttons available:

- `LV_KEY_ENTER`: will simulate press or pushing of the encoder button.
- `LV_KEY_LEFT`: will simulate turning encoder left.
- `LV_KEY_RIGHT`: will simulate turning encoder right.
- other keys will be passed to the focused widget.

If you hold the keys it will simulate an encoder advance with period specified in `indev_drv.long_press_repeat_time`.

```
lv_indev_set_type(indev, LV_INDEV_TYPE_ENCODER);
...
void encoder_with_keys_read(lv_indev_t * indev, lv_indev_data_t * data){
    data->key = last_key(); /* Get the last pressed or released key */
    /* use LV_KEY_ENTER for encoder press */
    if(key_pressed()) data->state = LV_INDEV_STATE_PRESSED;
    else {
        data->state = LV_INDEV_STATE_RELEASED;
        /* Optionally you can also use enc_diff, if you have encoder */
        data->enc_diff = enc_get_new_moves();
    }
}
```

## Hardware Button

A *Hardware Button* here is an external button (switch) typically next to the screen which is assigned to specific coordinates of the screen. If a button is pressed it will simulate the pressing on the assigned coordinate, similar to a touchpad.

To assign Hardware Buttons to coordinates use `lv_indev_set_button_points(my_indev, points_array)`. `points_array` should look like `const lv_point_t points_array[] = { {12, 30}, {60, 90}, ... }`

### Important

`points_array` cannot be allowed to go out of scope. Either declare it as a global variable or as a static variable inside a function.

```

lv_indev_set_type(indev, LV_INDEV_TYPE_BUTTON);
...

void button_read(lv_indev_t * indev, lv_indev_data_t * data){
    static uint32_t last_btn = 0; /* Store the last pressed button */
    int btn_pr = my_btn_read(); /* Get the ID (0,1,2...) of the pressed button */
    if(btn_pr >= 0) { /* Is there a button press? (E.g. -1 indicated no button was pressed) */
        last_btn = btn_pr; /* Save the ID of the pressed button */
        data->state = LV_INDEV_STATE_PRESSED; /* Set the pressed state */
    } else {
        data->state = LV_INDEV_STATE_RELEASED; /* Set the released state */
    }
    data->btn_id = last_btn; /* Save the last button */
}

```

When the `button_read` callback in the example above changes the `data->btn_id` to 0 a press/release action at the first index of the `points_array` will be performed (`{12,30}`).

## 7.2.2 Altre Funzionalità

### Parameters

The default value of the following parameters can be changed in `lv_indev_t`:

- `scroll_limit` Number of pixels to slide before actually scrolling the Widget
- `scroll_throw` Scroll throw (momentum) slow-down in [%]. Greater value means faster slow-down.
- `long_press_time` Press time to send `LV_EVENT_LONG_PRESSED` (in milliseconds)
- `long_press_repeat_time` Interval of sending `LV_EVENT_LONG_PRESSED_REPEAT` (in milliseconds)
- `read_timer` pointer to the `lv_timer` which reads the input device. Its parameters can be changed by calling `lv_timer_...()` functions. `LV_DEF_REFR_PERIOD` in `lv_conf.h` sets the default read period.

### Buffered Reading

By default, LVGL calls `read_cb` periodically. Because of this intermittent polling there is a chance that some user gestures are missed.

To solve this you can write an event driven driver for your input device that buffers measured data. In `read_cb` you can report the buffered data instead of directly reading the input device. Setting the `data->continue_reading` flag will tell LVGL there is more data to read and it should call `read_cb` again.

If the driver can provide precise timestamps for buffered events, it can overwrite `data->timestamp`. By default, this is initialized to `lv_tick_get()` just before invoking `read_cb`.

### Switching the Input Device to Event-Driven Mode

Normally an Input Device is read every `LV_DEF_REFR_PERIOD` milliseconds (set in `lv_conf.h`). However, in some cases, you might need more control over when to read the input device. For example, you might need to read it by polling a file descriptor (fd).

You can do this by:

```

/* Update the input device's running mode to LV_INDEV_MODE_EVENT */
lv_indev_set_mode(indev, LV_INDEV_MODE_EVENT);

...
/* Call this anywhere you want to read the input device */
lv_indev_read(indev);

```

#### Nota

`lv_indev_read()`, `lv_timer_handler()` and `_lv_display_refr_timer()` cannot run at the same time.

**i Nota**

For devices in event-driven mode, `data->continue_reading` is ignored.

**i Ulteriori Informazioni**

- `lv_port_indev_template.c` for a template for your own Input-Device driver.

### 7.2.3 API

`lv_indev.h`  
`lv_indev_gesture.h`  
`lv_indev_gesture_private.h`  
`lv_indev_private.h`  
`lv_indev_scroll.h`  
`lv_api_map_v8.h`  
`lv_api_map_v9_1.h`

## 7.3 Color (`lv_color`)

The color module handles all color-related functions like changing color depth, creating colors from hex code, converting between color depths, mixing colors, etc.

The type `lv_color_t` is used to store a color in RGB888 format. This type and format are used in almost all APIs regardless of `LV_COLOR_DEPTH`.

### 7.3.1 Creating Colors

#### RGB

Create colors from Red, Green and Blue channel values:

```
/* All channels are 0-255 */
lv_color_t c = lv_color_make(red, green, blue);

/* Same but can be used for const initialization as well */
lv_color_t c = LV_COLOR_MAKE(red, green, blue);

/* From hex code 0x000000..0xFFFFFFF interpreted as RED + GREEN + BLUE */
lv_color_t c = lv_color_hex(0x123456);

/* From 3 digits. Same as lv_color_hex(0x112233) */
lv_color_t c = lv_color_hex3(0x123);
```

#### HSV

Create colors from Hue, Saturation and Value values:

```
//h = 0..359, s = 0..100, v = 0..100
lv_color_t c = lv_color_hsv_to_rgb(h, s, v);

//All channels are 0-255
lv_color_hsv_t c_hsv = lv_color_rgb_to_hsv(r, g, b);

//From lv_color_t variable
lv_color_hsv_t c_hsv = lv_color_to_hsv(color);
```

## Palette

LVGL includes Material Design's palette of colors. In this system all named colors have a nominal main color as well as four darker and five lighter variants.

The names of the colors are as follows:

- LV\_PALETTE\_RED
- LV\_PALETTE\_PINK
- LV\_PALETTE\_PURPLE
- LV\_PALETTE\_DEEP\_PURPLE
- LV\_PALETTE\_INDIGO
- LV\_PALETTE\_BLUE
- LV\_PALETTE\_LIGHT\_BLUE
- LV\_PALETTE\_CYAN
- LV\_PALETTE\_TEAL
- LV\_PALETTE\_GREEN
- LV\_PALETTE\_LIGHT\_GREEN
- LV\_PALETTE\_LIME
- LV\_PALETTE\_YELLOW
- LV\_PALETTE\_AMBER
- LV\_PALETTE\_ORANGE
- LV\_PALETTE\_DEEP\_ORANGE
- LV\_PALETTE\_BROWN
- LV\_PALETTE\_BLUE\_GREY
- LV\_PALETTE\_GREY

To get the main color use `lv_color_t c = lv_palette_main(LV_PALETTE_...)`.

For the lighter variants of a palette color use `lv_color_t c = lv_palette_lighten(LV_PALETTE_..., v)`. `v` can be 1..5. For the darker variants of a palette color use `lv_color_t c = lv_palette_darken(LV_PALETTE_..., v)`. `v` can be 1..4.

## Modify and mix colors

The following functions can modify a color:

```
// Lighten a color. 0: no change, 255: white
lv_color_t c = lv_color_lighten(c, lvl);

// Darken a color. 0: no change, 255: black
lv_color_t c = lv_color_darken(lv_color_t c, lv_opa_t lvl);

// Lighten or darken a color. 0: black, 128: no change 255: white
lv_color_t c = lv_color_change_lightness(lv_color_t c, lv_opa_t lvl);

// Mix two colors with a given ratio 0: full c2, 255: full c1, 128: half c1 and half c2
lv_color_t c = lv_color_mix(c1, c2, ratio);
```

## Built-in colors

`lv_color_white()` and `lv_color_black()` return `0xFFFFFFFF` and `0x000000` respectively.

### 7.3.2 Opacity

To describe opacity the `lv_opa_t` type is created from `uint8_t`. Some special purpose defines are also introduced:

- `LV_OPA_TRANS` Value: 0, means no opacity making the color completely transparent
- `LV_OPA_10` Value: 25, means the color covers only a little
- `LV_OPA_20 ... OPA_80` follow logically
- `LV_OPA_90` Value: 229, means the color nearly completely covers
- `LV_OPA_COVER` Value: 255, means the color completely covers (full opacity)

You can also use the `LV_OPA_*` defines in `lv_color_mix()` as a mixing *ratio*.

### 7.3.3 API

`lv_draw_ppa_private.h`

`lv_color.h`

`lv_color_op.h`

## 7.4 Font (`lv_font`)

In LVGL fonts are collections of bitmaps and other information required to render images of individual letters (glyph). A font is stored in a `lv_font_t` variable and can be set in a style's `text_font` field. Ad esempio:

```
lv_style_set_text_font(&my_style, &lv_font_montserrat_28); /* Set a larger font */
```

Fonts have a **format** property. It describes how the glyph data is stored. At the time of writing, there are 12 possible values that this field can take, and those values fall into 2 categories:

#### Legacy simple

1, 2, 4 or 8-bpp (aligned or unaligned) and image format, and

#### Advanced

vector, SVG, and custom formats; for the latter, the user provides the rendering logic.

For simple formats:

- the font is stored as an array of bitmaps, one bitmap per glyph;
- the value stored for each pixel determines the pixel's opacity, enabling edges to be smoother --- higher bpp values result in smoother edges.

For advanced formats, the font information is stored in its respective format.

The **format** property also affects the amount of memory needed to store a font. For example, `format = LV_FONT_GLYPH_FORMAT_A4` makes a font nearly four times larger compared to `format = LV_FONT_GLYPH_FORMAT_A1`.

### 7.4.1 Unicode Support

LVGL supports **UTF-8** encoded Unicode characters. Your editor needs to be configured to save your code/text as UTF-8 (usually this is the default) and be sure that `LV_TXT_ENC` is set to `LV_TXT_ENC_UTF8` in `lv_conf.h`. (This is the default value.)

To test it try

```
lv_obj_t * label1 = lv_label_create(lv_screen_active(), NULL);
lv_label_set_text(label1, LV_SYMBOL_OK);
```

If all works well, a '✓' character should be displayed.

## 7.4.2 Typesetting

Although LVGL can decode and display any Unicode characters (assuming the font supports them), LVGL cannot correctly render all complex languages.

The standard Latin-based languages (e.g., English, Spanish, German) and East Asian languages such as Chinese, Japanese, and Korean (CJK) are relatively straightforward, as their characters are simply written from left to right.

Languages like Arabic, Persian, and Hebrew, which use Right-to-Left (RTL) or mixed writing directions, are also supported in LVGL. Learn more [here](#).

For characters such as 'é', 'ü', 'ö', 'Á', and 'Ü', it is recommended to use the single Unicode format (NFC) rather than decomposing them into a base letter and diacritics (e.g. u + '̄').

Complex languages where subsequent characters combine into a single glyph and where the resulting glyph has no individual Unicode representation (e.g., Devanagari), have limited support in LVGL.

## 7.4.3 Built-In Fonts

There are several built-in fonts in different sizes, which can be enabled in `lv_conf.h` with `LV_FONT_...` defines.

### Normal Fonts

The following fonts contain all ASCII characters, the degree symbol (U+00B0), the bullet symbol (U+2022) and the built-in symbols (see below).

- `LV_FONT_MONTserrat_12`: 12 px font
- `LV_FONT_MONTserrat_14`: 14 px font
- `LV_FONT_MONTserrat_16`: 16 px font
- `LV_FONT_MONTserrat_18`: 18 px font
- `LV_FONT_MONTserrat_20`: 20 px font
- `LV_FONT_MONTserrat_22`: 22 px font
- `LV_FONT_MONTserrat_24`: 24 px font
- `LV_FONT_MONTserrat_26`: 26 px font
- `LV_FONT_MONTserrat_28`: 28 px font
- `LV_FONT_MONTserrat_30`: 30 px font
- `LV_FONT_MONTserrat_32`: 32 px font
- `LV_FONT_MONTserrat_34`: 34 px font
- `LV_FONT_MONTserrat_36`: 36 px font
- `LV_FONT_MONTserrat_38`: 38 px font
- `LV_FONT_MONTserrat_40`: 40 px font
- `LV_FONT_MONTserrat_42`: 42 px font
- `LV_FONT_MONTserrat_44`: 44 px font
- `LV_FONT_MONTserrat_46`: 46 px font
- `LV_FONT_MONTserrat_48`: 48 px font

### Special fonts

- `LV_FONT_MONTserrat_28_COMPRESSED`: Same as normal 28 px font but stored as a [Compressed fonts](#) with 3 bpp
- `LV_FONT_DEJAVU_16_PERSIAN_HEBREW`: 16 px font with normal range + Hebrew, Arabic, Persian letters and all their forms

- *LV\_FONT\_SOURCE\_HAN\_SANS\_SC\_16\_CJK*: 16 px font with normal range plus 1000 of the most common CJK radicals
- *LV\_FONT\_UNSCII\_8*: 8 px pixel perfect font with only ASCII characters
- *LV\_FONT\_UNSCII\_16*: 16 px pixel perfect font with only ASCII characters

The built-in fonts are **global variables** with names like *lv\_font\_montserrat\_16* for a 16 px height font. To use them in a style, just add a pointer to a font variable like this:

```
lv_style_set_text_font(&my_style, &lv_font_montserrat_28);
```

The built-in fonts with **bpp = 4** contain the ASCII characters and use the **Montserrat** font.

In addition to the ASCII range, the following symbols are also added to the built-in fonts from the **FontAwesome** font.

 LV_SYMBOL_AUDIO	 LV_SYMBOL_WARNING
 LV_SYMBOL_VIDEO	 LV_SYMBOL_SHUFFLE
 LV_SYMBOL_LIST	 LV_SYMBOL_UP
 LV_SYMBOL_OK	 LV_SYMBOL_DOWN
 LV_SYMBOL_CLOSE	 LV_SYMBOL_LOOP
 LV_SYMBOL_POWER	 LV_SYMBOL_DIRECTORY
 LV_SYMBOL_SETTINGS	 LV_SYMBOL_UPLOAD
 LV_SYMBOL_TRASH	 LV_SYMBOL_CALL
 LV_SYMBOL_HOME	 LV_SYMBOL_CUT
 LV_SYMBOL_DOWNLOAD	 LV_SYMBOL_COPY
 LV_SYMBOL_DRIVE	 LV_SYMBOL_SAVE
 LV_SYMBOL_REFRESH	 LV_SYMBOL_CHARGE
 LV_SYMBOL_MUTE	 LV_SYMBOL_PASTE
 LV_SYMBOL_VOLUME_MID	 LV_SYMBOL_BELL
 LV_SYMBOL_VOLUME_MAX	 LV_SYMBOL_KEYBOARD
 LV_SYMBOL_IMAGE	 LV_SYMBOL_GPS
 LV_SYMBOL_EDIT	 LV_SYMBOL_FILE
 LV_SYMBOL_PREV	 LV_SYMBOL_WIFI
 LV_SYMBOL_PLAY	 LV_SYMBOL_BATTERY_FULL
 LV_SYMBOL_PAUSE	 LV_SYMBOL_BATTERY_3
 LV_SYMBOL_STOP	 LV_SYMBOL_BATTERY_2
 LV_SYMBOL_NEXT	 LV_SYMBOL_BATTERY_1
 LV_SYMBOL_EJECT	 LV_SYMBOL_BATTERY_EMPTY
 LV_SYMBOL_LEFT	 LV_SYMBOL_USB
 LV_SYMBOL_RIGHT	 LV_SYMBOL_BLUETOOTH
 LV_SYMBOL_PLUS	 LV_SYMBOL_BACKSPACE
 LV_SYMBOL_MINUS	 LV_SYMBOL_SD_CARD
 LV_SYMBOL_EYE_OPEN	 LV_SYMBOL_NEW_LINE
 LV_SYMBOL_EYE_CLOSE	

The symbols can be used singly as:

```
lv_label_set_text(my_label, LV_SYMBOL_OK);
```

Or together with strings (compile time string concatenation):

```
lv_label_set_text(my_label, LV_SYMBOL_OK "Apply");
```

Or more symbols together:

```
lv_label_set_text(my_label, LV_SYMBOL_OK LV_SYMBOL_WIFI LV_SYMBOL_PLAY);
```

## 7.4.4 Special Features

### Bidirectional support

Most languages use a Left-to-Right (LTR for short) writing direction, however some languages (such as Hebrew, Persian or Arabic) use Right-to-Left (RTL for short) direction.

LVGL not only supports RTL text but supports mixed (a.k.a. bidirectional, BiDi) text rendering as well. Alcuni esempi:

The names of these states in Arabic  
are الكويت and مصر respectively.

The title is مفتاح معايير الويب! in Arabic.

BiDi support is enabled by setting `LV_USE_BIDI` to a non-zero value in `lv_conf.h`.

All text has a base direction (LTR or RTL) which determines some rendering rules and the default alignment of the text (left or right). However, in LVGL, the base direction is not only applied to labels. It's a general property which can be set for every Widget. If not set then it will be inherited from the parent. This means it's enough to set the base direction of a screen and its child Widgets will inherit it.

The default base direction for screens can be set by `LV_BIDI_BASE_DIR_DEF` in `lv_conf.h` and other Widgets inherit the base direction from their parent.

To set a Widget's base direction use `lv_obj_set_style_base_dir(widget, base_dir, selector)`. The possible base directions are:

- `LV_BASE_DIR_LTR`: Left to Right base direction
- `LV_BASE_DIR_RTL`: Right to Left base direction
- `LV_BASE_DIR_AUTO`: Auto detect base direction

This list summarizes the effect of RTL base direction on Widgets:

- Create Widgets by default on the right
- `lv_tabview`: Displays tabs from right to left
- `lv_checkbox`: Shows the box on the right
- `lv_buttonmatrix`: Orders buttons from right to left
- `lv_list`: Shows icons on the right
- `lv_dropdown`: Aligns options to the right
- The text strings in `lv_table`, `lv_buttonmatrix`, `lv_keyboard`, `lv_tabview`, `lv_dropdown`, `lv_roller` are "BiDi processed" to be displayed correctly

## Arabic and Persian support

There are some special rules to display Arabic and Persian characters: the *form* of a character depends on its position in the text. A different form of the same letter needs to be used when it is isolated, at start, middle or end positions. Besides these, some conjunction rules should also be taken into account.

LVGL supports these rules if `LV_USE_ARABIC_PERSIAN_CHARS` is enabled in `lv_conf.h`.

However, there are some limitations:

- Only displaying text is supported (e.g. on labels), i.e. text inputs (e.g. Text Area) do not support this feature.
- Static text (i.e. const) is not processed. E.g. text set by `lv_label_set_text()` will be "Arabic processed" but `lv_label_set_text_static()` will not.
- Text get functions (e.g. `lv_label_get_text()`) will return the processed text.

## Compressed fonts

The built-in font engine supports compressed bitmaps. Compressed fonts can be generated by

- ticking the **Compressed** check box in the online converter
- not passing the `--no-compress` flag to the offline converter (compression is applied by default)

Compression is more effective with larger fonts and higher bpp. However, it's about 30% slower to render compressed fonts. Therefore, it is recommended to compress only the largest fonts of a user interface, because

- they need the most memory
- they can be compressed better
- and on the likelihood that they are used less frequently than the medium-sized fonts, the performance cost will be smaller.

Compressed fonts also support `bpp=3`.

## Kerning

Fonts may provide kerning information to adjust the spacing between specific characters.

- The online converter generates kerning tables.
- The offline converter generates kerning tables unless `--no-kerning` is specified.
- FreeType integration does not currently support kerning.
- The Tiny TTF font engine supports GPOS (Glyph Positioning) and Kern tables.

To configure kerning at runtime, use `lv_font_set_kerning()`.

### 7.4.5 Adding a New Font

There are several ways to add a new font to your project:

1. The simplest method is to use the [Online font converter](#). Just set the parameters, click the *Convert* button, copy the font to your project and use it. **Be sure to carefully read the steps provided on that site or you will get an error while converting.**
2. Use the [Offline font converter](#). (Requires Node.js to be installed)
3. If you want to create something like the built-in fonts (Montserrat font and symbols) but in a different size and/or ranges, you can use the `built_in_font_gen.py` script in `lvgl/scripts/built_in_font` folder. (This requires Python and [https://github.com/lvgl/lv\\_font\\_conv/](https://github.com/lvgl/lv_font_conv/) to be installed.)

To declare a font in a file, use `LV_FONT_DECLARE(my_font_name)`.

To make fonts globally available (like the built-in fonts), add them to `LV_FONT_CUSTOM_DECLARE` in `lv_conf.h`.

## 7.4.6 Adding New Symbols

The built-in symbols are created from the `FontAwesome` font.

1. Search for a symbol on <https://fontawesome.com>. For example the `USB` symbol. Copy its Unicode ID which is `0xf287`.
2. Open the [Online font converter](#). Add `FontAwesome.woff`.
3. Set the parameters such as Name, Size, BPP. You'll use this name to declare and use the font in your code.
4. Add the Unicode ID of the symbol to the range field. E.g. `0xf287` for the `USB` symbol. More symbols can be enumerated with `,`.
5. Convert the font and copy the generated source code to your project. Make sure to compile the `.c` file of your font.
6. Declare the font using `extern lv_font_t my_font_name;` or simply use `LV_FONT_DECLARE(my_font_name)`.

### Using the symbol

1. Convert the Unicode value to UTF8, for example on [this site](#). For `0xf287` the *Hex UTF-8 bytes* are `EF 8A 87`.
2. Create a `#define` string from the UTF8 values: `#define MY_USB_SYMBOL "\xEF\x8A\x87"`
3. Create a label and set the text. Eg. `lv_label_set_text(label, MY_USB_SYMBOL)`

#### note

`lv_label_set_text(label, MY_USB_SYMBOL)` searches for this symbol in the font defined in the style's `text.font` property. To use the symbol you will need to set the style's text font to use the generated font, e.g. `lv_style_set_text_font(&my_style, &my_font_name)` or `lv_obj_set_style_text_font(label, &my_font_name, 0)`.

## 7.4.7 Loading a Font at Run-Time

`lv_binfont_create()` can be used to load a font from a file. The font needs to have a special binary format. (Not TTF or WOFF). Use `lv_font_conv` with the `--format bin` option to generate an LVGL compatible font file.

#### note

To load a font *LVGL's filesystem* needs to be enabled and a driver must be added.

### Esempio

```
lv_font_t *my_font = lv_binfont_create("X:/path/to/my_font.bin");
if(my_font == NULL) return;
/* Use the font */
/* Free the font if not required anymore */
lv_binfont_destroy(my_font);
```

## 7.4.8 Loading a Font from a Memory Buffer at Run-Time

`lv_binfont_create_from_buffer()` can be used to load a font from a memory buffer. This function may be useful to load a font from an external file system, which is not supported by LVGL. The font needs to be in the same format as if it were loaded from a file.

#### note

To load a font from a buffer *LVGL's filesystem* needs to be enabled and the MEMFS driver must be added.

### Esempio

```
lv_font_t *my_font;
uint8_t *buf;
uint32_t bufsize;

/* Read font file into the buffer from the external file system */
...
```

(continues on next page)

(continua dalla pagina precedente)

```
/* Load font from the buffer */
my_font = lv_binfont_create_from_buffer((void *)buf, buf);
if(my_font == NULL) return;
/* Use the font */

/* Free the font if not required anymore */
lv_binfont_destroy(my_font);
```

## 7.4.9 Using a BDF Font

Small displays with low resolution don't look pretty with automatically rendered fonts. A bitmap font provides the solution, but it's necessary to convert the bitmap font (BDF) to a TTF.

### Convert BDF to TTF

BDF are bitmap fonts where fonts are not described in outlines but in pixels. BDF files can be used but they must be converted into the TTF format using `mkttf`, which can be found in this GitHub repository: <https://github.com/Tblue/mkttf>. This tool uses potrace to generate outlines from the bitmap information. The bitmap itself will be embedded into the TTF as well. `lv_font_conv` uses the embedded bitmap but it also needs the outlines. One might think you can use a fake MS Bitmap only sfnt (ttf) (TTF without outlines) created by fontforge, but this will not work.

Install imagemagick, python3, python3-fontforge and potrace

On Ubuntu Systems, just type

```
sudo apt install imagemagick python3-fontforge potrace
```

Clone mkttf

```
git clone https://github.com/Tblue/mkttf
```

Read the mkttf docs.

Former versions of imagemagick needs the imagemagick call in front of convert, identify and so on. But newer versions don't. So you might want to change 2 lines in `potrace-wrapper.sh` --- open `potrace-wrapper.sh` and remove imagemagick from line 55 and line 64:

line 55

```
wh=$(identify -format '%[width]pt %[height]pt' "${input?}")
```

line 64

```
convert "${input?}" -sample '1000%' - \
```

It might be necessary to change the mkttf.py script.

line 1

```
#!/usr/bin/env python3
```

### Example for a 12px font

```
cd mkttf
./mkttf.py ./TerminusMedium-12-12.bdf
Importing bitmaps from 0 additional fonts...
Importing font `./TerminusMedium-12-12.bdf' into glyph background...
Processing glyphs...
Saving TTF file...
Saving SFD file...
Done!
```

The TTF `TerminusMedium-001.000.ttf` will be created from `./TerminusMedium-12-12.bdf`.

To create a font for LVGL:

```
lv_font_conv --bpp 1 --size 12 --no-compress --font TerminusMedium-001.000.ttf --range 0x20-0x7e,0xa1-0xff --format lvgl -o terminus_1bpp_12px.c
```

#### note

use 1-bpp because we don't use anti-aliasing. It doesn't look sharp on displays with a low resolution.

## 7.4.10 Adding a New Font Engine

LVGL's font interface is designed to be very flexible but, even so, you can add your own font engine in place of LVGL's internal one. For example, you can use [FreeType](#) to real-time render glyphs from TTF fonts or use an external flash to store the font's bitmap and read them when the library needs them. FreeType can be used in LVGL as described in [Freetype](#).

To add a new font engine, a custom `lv_font_t` variable needs to be created:

```
/* Describe the properties of a font */
lv_font_t my_font;
my_font.get_glyph_dsc = my_get_glyph_dsc_cb;           /* Set a callback to get info about glyphs */
my_font.get_glyph_bitmap = my_get_glyph_bitmap_cb;      /* Set a callback to get bitmap of a glyph */
my_font.line_height = height;                          /* The real line height where any text fits */
my_font.base_line = base_line;                        /* Base line measured from the top of line_height */
my_font.dsc = something_required;                      /* Store any implementation specific data here */
my_font.user_data = user_data;                         /* Optionally some extra user data */

...
/* Get info about glyph of `unicode_letter` in `font` font.
 * Store the result in `dsc_out`.
 * The next letter (`unicode_letter_next`) might be used to calculate the width required by this glyph (kerning)
 */
bool my_get_glyph_dsc_cb(const lv_font_t * font, lv_font_glyph_dsc_t * dsc_out, uint32_t unicode_letter, uint32_t unicode_letter_next)
{
    /* Your code here */

    /* Store the result.
     * For example ...
     */
    dsc_out->adv_w = 12;          /* Horizontal space required by the glyph in [px] */
    dsc_out->box_h = 8;           /* Height of the bitmap in [px] */
    dsc_out->box_w = 6;           /* Width of the bitmap in [px] */
    dsc_out->ofs_x = 0;           /* X offset of the bitmap in [pf] */
    dsc_out->ofs_y = 3;           /* Y offset of the bitmap measured from the as line */
    dsc_out->format= LV_FONT_GLYPH_FORMAT_A2;

    return true;                  /* true: glyph found; false: glyph was not found */
}

/* Get the bitmap of `unicode_letter` from `font`. */
const uint8_t * my_get_glyph_bitmap_cb(const lv_font_t * font, uint32_t unicode_letter)
{
    /* Your code here */

    /* The bitmap should be a continuous bitstream where
     * each pixel is represented by `bpp` bits */
    return bitmap;    /* Or NULL if not found */
}
```

## 7.4.11 Using Font Fallback

If the font in use does not have a glyph needed in a text-rendering task, you can specify a **fallback** font to be used in `lv_font_t`.

**fallback** can be chained, so it will try to solve until there is no **fallback** set.

```
/* Roboto font doesn't have support for CJK glyphs */
lv_font_t *robot = my_font_load_function();
/* Droid Sans Fallback has more glyphs but its typeface doesn't look good as Roboto */
lv_font_t *droid_sans_fallback = my_font_load_function();
/* So now we can display Roboto for supported characters while having wider characters set support */
robot->fallback = droid_sans_fallback;
```

## 7.4.12 API

```
lv_font.h
lv_font_fmt_txt.h
lv_font_fmt_txt_private.h
lv_font_manager.h
lv_font_manager_recycle.h
```

## 7.5 Images (lv\_image)

An image can be a file or a variable that stores the bitmap itself and some metadata.

## 7.5.1 Store images

You can store images in two places

- as a variable in internal memory (RAM or ROM)
- as a file

### Variables

Images stored internally in a variable are composed mainly of an `lv_image_dsc_t` structure with the following fields:

- **header:**
  - *cf*: Color format. See *below*
  - *w*: width in pixels (<= 2048)
  - *h*: height in pixels (<= 2048)
  - *always zero*: 3 bits which need to be always zero
  - *reserved*: reserved for future use
- **data:** pointer to an array where the image itself is stored
- **data\_size:** length of **data** in bytes

These are usually stored within a project as C files. They are linked into the resulting executable like any other constant data.

### Files

To deal with files you need to add a storage *Drive* to LVGL. In short, a *Drive* is a collection of functions (*open*, *read*, *close*, etc.) registered in LVGL to make file operations. You can add an interface to a standard file system (FAT32 on SD card) or you create your simple file system to read data from an SPI Flash memory. In every case, a *Drive* is just an abstraction to read and/or write data to memory. See the *File system* section to learn more.

Images stored as files are not linked into the resulting executable, and must be read into RAM before being drawn. As a result, they are not as resource-friendly as images linked at compile time. However, they are easier to replace without needing to rebuild the main program.

## 7.5.2 Color formats

Various built-in color formats are supported:

- **`LV_COLOR_FORMAT_NATIVE`**: Simply stores the RGB colors (in whatever color depth LVGL is configured for).
- **`LV_COLOR_FORMAT_NATIVE_WITH_ALPHA`**: Like `LV_COLOR_FORMAT_NATIVE` but it also adds an alpha (transparency) byte for every pixel.
- **`LV_COLOR_FORMAT_I1`, `LV_COLOR_FORMAT_I2`, `LV_COLOR_FORMAT_I4`, `LV_COLOR_FORMAT_I8`**: Uses a palette with 2, 4, 16 or 256 colors and stores each pixel in 1, 2, 4 or 8 bits.

Since the colors in the palette can be anything, the pixel data for these formats is stored as ARGB8888.

When using LVGL's SW render, make sure `LV_DRAW_SW_SUPPORT_ARGB8888` is set to 1 in your LVGL config.

- **`LV_COLOR_FORMAT_A1`, `LV_COLOR_FORMAT_A2`, `LV_COLOR_FORMAT_A4`, `LV_COLOR_FORMAT_A8`**: Only stores the Alpha value with 1, 2, 4 or 8 bits. The pixels take the color of `style`.`img_recolor` and the set opacity. The source image has to be an alpha channel. This is ideal for bitmaps similar to fonts where the whole image is one color that can be altered.

The bytes of `LV_COLOR_FORMAT_NATIVE` images are stored in the following order.

- 32-bit color depth:
  - **Byte 0:** Blue
  - **Byte 1:** Green
  - **Byte 2:** Red
  - **Byte 3:** Alpha (only with `LV_COLOR_FORMAT_NATIVE_WITH_ALPHA`)
- 16-bit color depth:
  - **Byte 0:** Green 3 lower bit, Blue 5 bit
  - **Byte 1:** Red 5 bit, Green 3 higher bit
  - **Byte 2:** Alpha byte (only with `LV_COLOR_FORMAT_NATIVE_WITH_ALPHA`)
- 8-bit color depth:
  - **Byte 0:** Red 3 bit, Green 3 bit, Blue 2 bit
  - **Byte 2:** Alpha byte (only with `LV_COLOR_FORMAT_NATIVE_WITH_ALPHA`)

You can store images in a *Raw* format to indicate that it's not encoded with one of the built-in color formats and an external *Image decoder* needs to be used to decode the image.

- `LV_COLOR_FORMAT_RAW`: Indicates a basic raw image (e.g. a PNG or JPG image).
- `LV_COLOR_FORMAT_RAW_ALPHA`: Indicates that an image has alpha and an alpha byte is added for every pixel.

### 7.5.3 Add and use images

You can add images to LVGL in two ways:

- using the online converter
- manually create images

#### Online converter

The online Image converter is available here: <https://lvgl.io/tools/imageconverter>

Adding an image to LVGL via the online converter is easy.

1. You need to select a *BMP*, *PNG* or *JPG* image first.
2. Give the image a name that will be used within LVGL.
3. Select the *Color format*.
4. Select the type of image you want. Choosing a binary will generate a `.bin` file that must be stored separately and read using the *file support*. Choosing a variable will generate a standard C file that can be linked into your project.
5. Hit the *Convert* button. Once the conversion is finished, your browser will automatically download the resulting file.

In the generated C arrays (variables), bitmaps for all the color depths (1, 8, 16 or 32) are included in the C file, but only the color depth that matches `LV_COLOR_DEPTH` in `lv_conf.h` will actually be linked into the resulting executable.

In the case of binary files, you need to specify the color format you want:

- RGB332 for 8-bit color depth
- RGB565 for 16-bit color depth
- RGB565 Swap for 16-bit color depth (two bytes are swapped)
- RGB888 for 32-bit color depth

## Manually create an image

If you are generating an image at run-time, you can craft an image variable to display it using LVGL. Ad esempio:

```
uint8_t my_img_data[] = {0x00, 0x01, 0x02, ...};

static lv_image_dsc_t my_img_dsc = {
    .header.always_zero = 0,
    .header.w = 80,
    .header.h = 60,
    .data_size = 80 * 60 * LV_COLOR_DEPTH / 8,
    .header.cf = LV_COLOR_FORMAT_NATIVE, /* Set the color format */
    .data = my_img_data,
};
```

Another (possibly simpler) option to create and display an image at run-time is to use the *Canvas* Widget.

## Use images

The simplest way to use an image in LVGL is to display it with an *Image (lv\_image)* Widget:

```
lv_obj_t * icon = lv_image_create(lv_screen_active(), NULL);

/* From variable */
lv_image_set_src(icon, &my_icon_dsc);

/* From file */
lv_image_set_src(icon, "S:my_icon.bin");
```

If the image was converted with the online converter, you should use `LV_IMAGE_DECLARE(my_icon_dsc)` to declare the image in the file where you want to use it.

### 7.5.4 Image decoder

As you can see in the *Color formats* section, LVGL supports several built-in image formats. In many cases, these will be all you need. LVGL doesn't directly support, however, generic image formats like PNG or JPG.

To handle non-built-in image formats, you need to use external libraries and attach them to LVGL via the *Image decoder* interface.

An image decoder consists of 4 callbacks:

#### info

get some basic info about the image (width, height and color format).

#### open

open an image: - store a decoded image - set it to `NULL` to indicate the image can be read line-by-line.

#### get\_area

if `open` didn't fully open an image this function should give back part of image as decoded data.

#### close

close an opened image, free the allocated resources.

You can add any number of image decoders. When an image needs to be drawn, the library will try all the registered image decoders until it finds one which can open the image, i.e. one which knows that format.

The following formats are understood by the built-in decoder:

- `LV_COLOR_FORMAT_I1`
- `LV_COLOR_FORMAT_I2`
- `LV_COLOR_FORMAT_I4`
- `LV_COLOR_FORMAT_I8`
- `LV_COLOR_FORMAT_RGB888`
- `LV_COLOR_FORMAT_XRGB8888`
- `LV_COLOR_FORMAT_ARGB8888`
- `LV_COLOR_FORMAT_RGB565`

- `LV_COLOR_FORMAT_RGB565A8`

## Custom image formats

The easiest way to create a custom image is to use the online image converter and select **Raw** or **Raw with alpha** format. It will just take every byte of the binary file you uploaded and write it as an image "bitmap". You then need to attach an image decoder that will parse that bitmap and generate the real, renderable bitmap.

`header.cf` will be `LV_COLOR_FORMAT_RAW`, `LV_COLOR_FORMAT_RAW_ALPHA` accordingly. You should choose the correct format according to your needs: a fully opaque image, using an alpha channel.

After decoding, the *raw* formats are considered *True color* by the library. In other words, the image decoder must decode the *Raw* images to *True color* according to the format described in the *Color formats* section.

## Registering an image decoder

Here's an example of getting LVGL to work with PNG images.

First, you need to create a new image decoder and set some functions to open/close the PNG files. It should look like this:

```
/* Create a new decoder and register functions */
lv_image_decoder_t * dec = lv_image_decoder_create();
lv_image_decoder_set_info_cb(dec, decoder_info);
lv_image_decoder_set_open_cb(dec, decoder_open);
lv_image_decoder_set_get_area_cb(dec, decoder_get_area);
lv_image_decoder_set_close_cb(dec, decoder_close);

/***
 * Get info about a PNG image
 * @param decoder pointer to the decoder where this function belongs
 * @param src can be file name or pointer to a C array
 * @param header image information is set in header parameter
 * @return LV_RESULT_OK: no error; LV_RESULT_INVALID: can't get the info
 */
static lv_result_t decoder_info(lv_image_decoder_t * decoder, const void * src, lv_image_header_t * header)
{
    /* Check whether the type `src` is known by the decoder */
    if(is_png(src) == false) return LV_RESULT_INVALID;

    /* Read the PNG header and find `width` and `height` */
    ...

    header->cf = LV_COLOR_FORMAT_ARGB8888;
    header->w = width;
    header->h = height;
}

/***
 * Open a PNG image and decode it into dsc.decoded
 * @param decoder pointer to the decoder where this function belongs
 * @param dsc image descriptor
 * @return LV_RESULT_OK: no error; LV_RESULT_INVALID: can't open the image
 */
static lv_result_t decoder_open(lv_image_decoder_t * decoder, lv_image_decoder_dsc_t * dsc)
{
    (void) decoder; /* Unused */

    /* Check whether the type `src` is known by the decoder */
    if(is_png(dsc->src) == false) return LV_RESULT_INVALID;

    /* Decode and store the image. If `dsc->decoded` is `NULL`, the `decoder_get_area` function will be called to get the image data line-by-line */
    dsc->decoded = my_png_decoder(dsc->src);

    /* Change the color format if decoded image format is different than original format. For PNG it's usually decoded to ARGB8888 format */
    dsc->decoded.header.cf = LV_COLOR_FORMAT_...

    /* Call a binary image decoder function if required. It's not required if `my_png_decoder` opened the image in true color format. */
    lv_result_t res = lv_bin_decoder_open(decoder, dsc);

    return res;
}

/***
 * Decode an area of image
 * @param decoder pointer to the decoder where this function belongs
 * @param dsc image decoder descriptor
 * @param full_area input parameter. the full area to decode after enough subsequent calls
 * @param decoded_area input+output parameter. set the values to `LV_COORD_MIN` for the first call and to reset decoding.
 * @param return LV_RESULT_OK: ok; LV_RESULT_INVALID: failed or there is nothing left to decode
 */
static lv_result_t decoder_get_area(lv_image_decoder_t * decoder, lv_image_decoder_dsc_t * dsc,
                                    const lv_area_t * full_area, lv_area_t * decoded_area)
{
    /* If `dsc->decoded` is always set in `decoder_open` then `decoder_get_area` does not need to be implemented.
     * If `dsc->decoded` is only sometimes set or never set in `decoder_open` then `decoder_get_area` is used to
     * incrementally decode the image by calling it repeatedly until it returns `LV_RESULT_INVALID`.
     * In the example below the image is decoded line-by-line but the decoded area can have any shape and size
     * depending on the requirements and capabilities of the image decoder.
     */

    my_decoder_data_t * my_decoder_data = dsc->user_data;

    /* if `decoded_area` has a field set to `LV_COORD_MIN` then reset decoding */
    if(decoded_area->y1 == LV_COORD_MIN) {
```

(continues on next page)

(continua dalla pagina precedente)

```

decoded_area->x1 = full_area->x1;
decoded_area->x2 = full_area->x2;
decoded_area->y1 = full_area->y1;
decoded_area->y2 = decoded_area->y1; /* decode line-by-line, starting with the first line */

/* create a draw buf the size of one line */
bool reshape_success = NULL != lv_draw_buf_reshape(my_decoder_data->partial,
                                                    dsc->decoded.header.cf,
                                                    lv_area_get_width(full_area),
                                                    1,
                                                    LV_STRIDE_AUTO);

if(!reshape_success) {
    lv_draw_buf_destroy(my_decoder_data);
    my_decoder_data->partial = lv_draw_buf_create(lv_area_get_width(full_area),
                                                    1,
                                                    dsc->decoded.header.cf,
                                                    LV_STRIDE_AUTO);

    my_png_decode_line_reset(full_area);
}
/* otherwise decoding is already in progress. decode the next line */
else {
    /* all lines have already been decoded. indicate completion by returning `LV_RESULT_INVALID` */
    if (decoded_area->y1 >= full_area->y2) return LV_RESULT_INVALID;
    decoded_area->y1++;
    decoded_area->y2++;
}

my_png_decode_line(my_decoder_data->partial);

return LV_RESULT_OK;
}

/**
 * Close PNG image and free data
 * @param decoder pointer to the decoder where this function belongs
 * @param dsc image decoder descriptor
 * @return LV_RESULT_OK: no error; LV_RESULT_INVALID: can't open the image
 */
static void decoder_close(lv_image_decoder_t * decoder, lv_image_decoder_dsc_t * dsc)
{
    /* Free all allocated data */
    my_png_cleanup();

    my_decoder_data_t * my_decoder_data = dsc->user_data;
    lv_draw_buf_destroy(my_decoder_data->partial);

    /* Call the built-in close function if the built-in open/get_area was used */
    lv_bin_decoder_close(decoder, dsc);
}

```

So in summary:

- In `decoder_info`, you should collect some basic information about the image and store it in `header`.
- In `decoder_open`, you should try to open the image source pointed by `dsc->src`. Its type is already in `dsc->src_type == LV_IMG_SRC_FILE/VARIABLE`. If this format/type is not supported by the decoder, return `LV_RESULT_INVALID`. However, if you can open the image, a pointer to the decoded image should be set in `dsc->decoded`. If the format is known, but you don't want to decode the entire image (e.g. no memory for it), set `dsc->decoded = NULL` and use `decoder_get_area` to get the image area pixels.
- In `decoder_close` you should free all allocated resources.
- `decoder_get_area` is optional. In this case you should decode the whole image In `decoder_open` function and store image data in `dsc->decoded`. Decoding the whole image requires extra memory and some computational overhead.

## Manually use an image decoder

LVGL will use registered image decoders automatically if you try and draw a raw image (i.e. using the `lv_image` Widget) but you can use them manually as well. Create an `lv_image_decoder_dsc_t` variable to describe the decoding session and call `lv_image_decoder_open()`.

The `color` parameter is used only with `LV_COLOR_FORMAT_A1/2/4/8` images to tell color of the image.

```

lv_result_t res;
lv_image_decoder_dsc_t dsc;
lv_image_decoder_args_t args = { 0 }; /* Custom decoder behavior via args */
res = lv_image_decoder_open(&dsc, &my_img_dsc, &args);

if(res == LV_RESULT_OK) {
    /* Do something with `dsc->decoded`. You can copy out the decoded image by `lv_draw_buf_dup(dsc.decoded)` */
    lv_image_decoder_close(&dsc);
}

```

## Image post-processing

Considering that some hardware has special requirements for image formats, such as alpha premultiplication and stride alignment, most image decoders (such as PNG decoders) may not directly output image data that meets hardware requirements.

For this reason, LVGL provides a solution for image post-processing. First, call a custom post-processing function after `lv_image_decoder_open` to adjust the data in the image cache, and then mark the processing status in `cache_entry->process_state` (to avoid repeated post-processing).

See the detailed code below:

- Stride alignment and premultiply post-processing example:

```
/* Define post-processing state */
typedef enum {
    IMAGE_PROCESS_STATE_NONE = 0,
    IMAGE_PROCESS_STATE_STRIDE_ALIGNED = 1 << 0,
    IMAGE_PROCESS_STATE_PREMULTIPLIED_ALPHA = 1 << 1,
} image_process_state_t;

lv_result_t my_image_post_process(lv_image_decoder_dsc_t * dsc)
{
    lv_color_format_t color_format = dsc->header.cf;
    lv_result_t res = LV_RESULT_OK;

    if(color_format == LV_COLOR_FORMAT_ARGB8888) {
        lv_cache_lock();
        lv_cache_entry_t * entry = dsc->cache_entry;

        if(!(entry->process_state & IMAGE_PROCESS_STATE_PREMULTIPLIED_ALPHA)) {
            lv_draw_buf_premultiply(dsc->decoded);
            LV_LOG_USER("premultiplied alpha OK");

            entry->process_state |= IMAGE_PROCESS_STATE_PREMULTIPLIED_ALPHA;
        }

        if(!(entry->process_state & IMAGE_PROCESS_STATE_STRIDE_ALIGNED)) {
            uint32_t stride_expect = lv_draw_buf_width_to_stride(decoded->header.w, decoded->header.cf);
            if(decoded->header.stride != stride_expect) {
                LV_LOG_WARN("Stride mismatch");
                lv_draw_buf_t * aligned = lv_draw_buf_adjust_stride(decoded, stride_expect);
                if(aligned == NULL) {
                    LV_LOG_ERROR("No memory for Stride adjust.");
                    return NULL;
                }

                decoded = aligned;
            }

            entry->process_state |= IMAGE_PROCESS_STATE_STRIDE_ALIGNED;
        }
    }

    alloc_failed:
    lv_cache_unlock();
}

return res;
}
```

- GPU draw unit example:

```
void gpu_draw_image(lv_draw_unit_t * draw_unit, const lv_draw_image_dsc_t * draw_dsc, const lv_area_t * coords)
{
    ...
    lv_image_decoder_dsc_t decoder_dsc;
    lv_result_t res = lv_image_decoder_open(&decoder_dsc, draw_dsc->src, NULL);
    if(res != LV_RESULT_OK) {
        LV_LOG_ERROR("Failed to open image");
        return;
    }

    res = my_image_post_process(&decoder_dsc);
    if(res != LV_RESULT_OK) {
        LV_LOG_ERROR("Failed to post-process image");
        return;
    }
    ...
}
```

## 7.5.5 Image caching

Sometimes it takes a lot of time to open an image. Continuously decoding a PNG/JPEG image or loading images from a slow external memory would be inefficient and detrimental to the user experience.

Therefore, LVGL caches image data. Caching means some images will be left open, hence LVGL can quickly access them from `dsc->decoded` instead of needing to decode them again.

Of course, caching images is resource intensive as it uses more RAM to store the decoded image. LVGL tries to optimize the process as much as possible (see below), but you will still need to evaluate if this would be beneficial for your platform or not. Image caching may not be worth it if you have a deeply embedded target which decodes small images from a relatively fast storage medium.

## Cache size

The size of cache (in bytes) can be defined with `LV_CACHE_DEF_SIZE` in `lv_conf.h`. The default value is 0, so no image is cached.

The size of cache can be changed at run-time with `lv_cache_set_max_size(size_t size)`, and get with `lv_cache_get_max_size()`.

## Value of images

When you use more images than available cache size, LVGL can't cache all the images. Instead, the library will close one of the cached images to free space.

To decide which image to close, LVGL uses a measurement it previously made of how long it took to open the image. Cache entries that hold slower-to-open images are considered more valuable and are kept in the cache as long as possible.

If you want or need to override LVGL's measurement, you can manually set the `weight` value in the cache entry in `cache_entry->weight = time_ms` to give a higher or lower value. (Leave it unchanged to let LVGL control it.)

Every cache entry has a "*life*" value. Every time an image is opened through the cache, the *life* value of all entries is increased by their *weight* values to make them older. When a cached image is used, its *usage\_count* value is increased to make it more alive.

If there is no more space in the cache, the entry with *usage\_count* == 0 and lowest life value will be dropped.

## Memory usage

Note that a cached image might continuously consume memory. For example, if three PNG images are cached, they will consume memory while they are open.

Therefore, it's the user's responsibility to be sure there is enough RAM to cache even the largest images at the same time.

## Clean the cache

Let's say you have loaded a PNG image into a `lv_image_dsc_t my_png` variable and use it in an `lv_image` Widget. If the image is already cached and you then change the underlying PNG file, you need to notify LVGL to cache the image again. Otherwise, there is no easy way of detecting that the underlying file changed and LVGL will still draw the old image from cache.

To do this, use `lv_cache_invalidate(lv_cache_find(&my_png, LV_CACHE_SRC_TYPE_PTR, 0, 0))`.

## Custom cache algorithm

If you want to implement your own cache algorithm, you can refer to the following code to replace the LVGL built-in cache manager:

```
static lv_cache_entry_t * my_cache_add_cb(size_t size)
{
    ...
}

static lv_cache_entry_t * my_cache_find_cb(const void * src, lv_cache_src_type_t src_type, uint32_t param1, uint32_t param2)
{
    ...
}

static void my_cache_invalidate_cb(lv_cache_entry_t * entry)
{
    ...
}

static const void * my_cache_get_data_cb(lv_cache_entry_t * entry)
{
    ...
}

static void my_cache_release_cb(lv_cache_entry_t * entry)
{
    ...
}
```

(continues on next page)

(continua dalla pagina precedente)

```

static void my_cache_set_max_size_cb(size_t new_size)
{
    ...
}

static void my_cache_empty_cb(void)
{
    ...
}

void my_cache_init(void)
{
    /* Initialize new cache manager. */
    lv_cache_manager_t my_manager;
    my_manager.add_cb = my_cache_add_cb;
    my_manager.find_cb = my_cache_find_cb;
    my_manager.invalidate_cb = my_cache_invalidate_cb;
    my_manager.get_data_cb = my_cache_get_data_cb;
    my_manager.release_cb = my_cache_release_cb;
    my_manager.set_max_size_cb = my_cache_set_max_size_cb;
    my_manager.empty_cb = my_cache_empty_cb;

    /* Replace existing cache manager with the new one. */
    lv_cache_lock();
    lv_cache_set_manager(&my_manager);
    lv_cache_unlock();
}

```

## 7.5.6 API

*lv\_draw\_buf.h*  
*lv\_draw\_image.h*  
*lv\_draw\_image\_private.h*  
*lv\_image\_decoder.h*  
*lv\_image\_decoder\_private.h*  
*lv\_image\_dsc.h*  
*lv\_api\_map\_v8.h*  
*lv\_api\_map\_v9\_0.h*  
*lv\_api\_map\_v9\_1.h*  
*lv\_image\_cache.h*  
*lv\_image\_header\_cache.h*  
*lv\_image.h*  
*lv\_image\_private.h*  
*lv\_obj\_property\_names.h*

## 7.6 Timer (*lv\_timer*)

LVGL has a built-in Timer system. You can register a function to have it be called periodically. The Timers are handled and called in *lv\_timer\_handler()*, which needs to be called every few milliseconds. See *Gestione del Timer* for more information.

By default, LVGL itself uses Timers to:

- refresh each display --- during the creation of each *Display* (*lv\_display*), a Timer is created for that Display. That Timer refreshes the display based on the configured value of *LV\_DEF\_REFR\_PERIOD*, and also sends all display-related events, like *LV\_EVENT\_REFR\_START*, *LV\_EVENT\_REFR\_READY*, etc.
- read input devices --- during the creation of each *Input Device* (*lv\_indev*), a Timer is created for that Input Device based on the configured value of *LV\_DEF\_REFR\_PERIOD*. That Timer causes that input device to be read and also sends all input-device-related events, like *LV\_EVENT\_CLICKED*, *LV\_EVENT\_PRESSED*, etc.
- update system-monitor values --- if *LV\_USE\_SYSMON* is set to 1 in *lv\_conf.h*, one or more timers are created to periodically compute and monitor system performance statistics and LVGL's memory usage.

Timers are non-preemptive, which means a Timer cannot interrupt another Timer. Therefore, you can call any LVGL-related function in a Timer.

### 7.6.1 Creating a Timer

To create a new Timer, use `lv_timer_create(timer_cb, period_ms, user_data)`. It returns an `lv_timer_t *` which can be used later to modify the parameters of the Timer, pause it, or delete it when it is no longer needed. `lv_timer_create_basic()` can also be used to create a new Timer without specifying any parameters.

A Timer callback should have this prototype: `void (*lv_timer_cb_t)(lv_timer_t *)`.

Ad esempio:

```
void my_timer(lv_timer_t * timer)
{
    /* Use the user_data */
    uint32_t * user_data = lv_timer_get_user_data(timer);
    printf("my_timer called with user data: %d\n", *user_data);

    /* Do something with LVGL */
    if(something_happened) {
        something_happened = false;
        lv_button_create(lv_screen_active());
    }
}

...
static uint32_t user_data = 10;
lv_timer_t * timer = lv_timer_create(my_timer, 500, &user_data);
```

### 7.6.2 Ready and Reset

`lv_timer_ready(timer)` makes a Timer run on the next call of `lv_timer_handler()`.

`lv_timer_reset(timer)` resets the period of a Timer. It will be called again after its currently-set period (in milliseconds) has elapsed.

See what happens when *no timers are ready* and `lv_timer_handler()` is called.

### 7.6.3 Setting Parameters

You can modify these Timer parameters at any time during its life:

- `lv_timer_set_cb(timer, new_cb)`
- `lv_timer_set_period(timer, new_period_ms)`
- `lv_timer_set_user_data(timer, user_data)`

### 7.6.4 Repeat Count

When a Timer is created, its repeat-count is set to -1 to cause it to repeat indefinitely. You can make a Timer repeat only a given number of times with `lv_timer_set_repeat_count(timer, count)`. By default, once the Timer has run `count` times, it will be automatically deleted.

You can use `lv_timer_set_auto_delete(timer, false)` if you want the timer to instead be paused after it has run `count` times. This can be handy if you reuse that timer repeatedly and want to avoid the CPU and `lv_malloc()` overhead of repeatedly creating and deleting a timer. If you use this option, you will need to set its repeat count (to either -1 or a positive repeat count, since it will have decremented to 0) and `resume` it to make it active again.

### 7.6.5 Pause and Resume

`lv_timer_pause(timer)` pauses the specified Timer.

`lv_timer_resume(timer)` resumes the specified Timer.

## 7.6.6 Measuring Idle Time

You can get the idle percentage time of `lv_timer_handler()` with `lv_timer_get_idle()`. Note that it does not measure the idle time of the overall system, only of `lv_timer_handler()`. This can be misleading if you are using an operating system and DMA and/or GPU are used during rendering, as it does not actually measure the time the OS spends in an idle thread.

If you are using an OS and wish to get the time the CPU is spending in an idle thread, one way of doing so is configuring `LV_USE_SYSMON` and `LV_USE_PERF_MONITOR` to 1 in `lv.conf.h` (if they are not already), and setting the macro `LV_SYSMON_GET_IDLE` to the name of a function that fetches the percent of CPU time spent in the OS's idle thread. An example of such a function is `lv_os_get_idle_percent()` in `lv_freertos.c`. While the configuration is set this way, some system performance statistics (including CPU load) will appear on the display in a partially-transparent label whose location is set by the `LV_USE_PERF_MONITOR_POS` macro.

## 7.6.7 Enable and Disable

You can temporarily disable Timer handling with `lv_timer_enable(false)`. Be advised: this also pauses handling of Timers that refresh Display(s) and read from input devices, so don't forget to re-enable it with `lv_timer_enable(true)` as soon as the need for the pause is over.

## 7.6.8 Timer Handler Resume Callback

When the Timer system has been disabled (causing `lv_timer_handler()` to return early before it has processed any timers), if you want to take some action when the Timer system is re-enabled again, set a resume callback using `lv_timer_handler_set_resume_cb(cb, user_data)`. The callback should have this prototype:

```
void (*lv_timer_handler_resume_cb_t)(void*)
```

## 7.6.9 Asynchronous calls

There are several cases in which you may not want to perform an action immediately. Some examples are:

- you cannot delete a Widget because something else is still using it,
- you don't want to block execution now, or
- you detect the need to delete a Widget in a thread other than the thread making LVGL calls (e.g. in a case where you are using a *Gateway Thread* to make all LVGL calls in a multi-threaded environment).

For these cases, `lv_async_call(my_function, data_p)` can be used to call `my_function` on the next invocation of `lv_timer_handler()`. As a side effect, this also ensures it is called in a thread in which it is safe to make LVGL calls. `data_p` will be passed to the function when it's called. Note that only the data's pointer is saved, so whatever it is pointing to needs to remain valid until the function is called, so it can point to `static`, global or dynamically allocated data. If you want to cancel an asynchronous call, call `lv_async_call_cancel(my_function, data_p)`, which will remove all asynchronous calls matching `my_function` and `data_p`.

Note that if `lv_async_call(my_function, data_p)` is called from a thread other than the one that normally makes LVGL calls, you are still obligated to protect the LVGL data structure using a MUXEX.

Ad esempio:

```
void my_screen_clean_up(void * scr)
{
    /* Free some resources related to `scr`*/
    /* Finally delete the screen */
    lv_obj_delete(scr);
}

...

/* Do something with the Widget on the current screen */

/* Delete screen on next call of `lv_timer_handler`, not right now. */
lv_lock();
lv_async_call(my_screen_clean_up, lv_screen_active());
lv_unlock();

/* The screen is still valid so you can do other things with it */
```

If you just want to delete a Widget and don't need to clean anything up in `my_screen_cleanup` you could just use `lv_obj_delete_async(widget)` which will delete the Widget on the next call to `lv_timer_handler()`.

### 7.6.10 API

`lv_api_map_v8.h`

`lv_timer.h`

`lv_timer_private.h`

## 7.7 Animation (lv\_anim)

Animations allow you to define the way something should move or change over time, and let LVGL do the heavy lifting of making it happen. What makes it so powerful is that the thing being changed can be virtually anything in your system. It is very convenient to apply this to LVGL Widgets in your user interface (UI), to change their appearance, size or location over time. But because it is --- at its core --- a generic change-over-time manager, complete with a variety of optional event callbacks, its application can be wider than just to UI components.

For each Animation you create, it accomplishes the above by providing a generic method of varying a signed integer from a start value to an end value over a specified time period. It allows you to specify what object it applies to (the "variable"), which is available in the callback functions that are called as the Animation is playing through.

This variation over time can be linear (default), it can be on a path (curve) that you specify, and there is even a variety of commonly-used non-linear effects that can be specified.

The main callback called during an Animation (when it is playing) is called an *animator* function, which has the following prototype:

```
void func(void *var, int32_t value);
```

This prototype makes it easy to use most of the LVGL *set* functions directly or via a trivial wrapper. It includes:

- most of the widget properties
- functions that set *local style properties* directly on objects (needs a wrapper to set the *selector*)
- set properties on `lv_style_t` objects (e.g. *shared styles*) (`lv_obj_report_style_change` needs to be called to notify the widgets having the style)
- `lv_style_set_<property_name>(&style, <value>)`
- `lv_obj_set_<property_name>(widget, <value>)`

Because of the former, an animation on a single `lv_style_t` object shared among several objects can simultaneously modify the appearance of all objects that use it. See *Stili* for more details.

Examples of the latter are: `lv_obj_set_x(widget, value)` or `lv_obj_set_width(widget, value)`.

This makes it very convenient to apply to the appearance (and other attributes) of UI components. But you can provide your own "set" functions, and so the application of Animations is really limited only by your imagination.

The number of Animations that can be playing at the same time for a given object with a given *animator* callback is one (1). However, the number of Animations that can be playing at the same time is limited only by available RAM and CPU time for:

- a given object with different *animator* callbacks; and
- different objects.

Thus, you can have a Button's width being changed by one Animation while having its height being changed by another Animation.

## 7.7.1 Create an Animation

To create an Animation, start by creating an Animation *template* in an `lv_anim_t` variable. It has to be initialized and configured with `lv_anim_set_...` functions.

```
/* INITIALIZE AN ANIMATION
*-----*/
static lv_anim_t anim_template;
static lv_anim_t * running_anim;

lv_anim_init(&anim_template);

/* MANDATORY SETTINGS
*-----*/
/* Set the "animator" function */
lv_anim_set_exec_cb(&anim_template, (lv_anim_exec_xcb_t) lv_obj_set_x);

/* Set target of the Animation */
lv_anim_set_var(&anim_template, widget);

/* Length of the Animation [ms] */
lv_anim_set_duration(&anim_template, duration_in_ms);

/* Set start and end values. E.g. 0, 150 */
lv_anim_set_values(&anim_template, start, end);

/* OPTIONAL SETTINGS
*-----*/
/* Time to wait before starting the Animation [ms] */
lv_anim_set_delay(&anim_template, delay);

/* Set path (curve). Default is linear */
lv_anim_set_path_cb(&anim_template, lv_anim_path_ease_in);

/* Set anim_template callback to indicate when the Animation is completed. */
lv_anim_set_completed_cb(&anim_template, completed_cb);

/* Set anim_template callback to indicate when the Animation is deleted (idle). */
lv_anim_set_deleted_cb(&anim_template, deleted_cb);

/* Set anim_template callback to indicate when the Animation is started (after delay). */
lv_anim_set_start_cb(&anim_template, start_cb);

/* When ready, play the Animation backward with this duration. Default is 0 (disabled) [ms] */
lv_anim_set_reverse_duration(&anim_template, time);

/* Delay before reverse play. Default is 0 (disabled) [ms] */
lv_anim_set_reverse_delay(&anim_template, delay);

/* Number of repetitions. Default is 1. LV_ANM_REPEAT_INFINITE for infinite repetition */
lv_anim_set_repeat_count(&anim_template, cnt);

/* Delay before repeat. Default is 0 (disabled) [ms] */
lv_anim_set_repeat_delay(&anim_template, delay);

/* true (default): apply the start value immediately, false: apply start value after delay when the Anim. really starts. */
lv_anim_set_early_apply(&anim_template, true/false);

/* START THE ANIMATION
*-----*/
running_anim = lv_anim_start(&anim_template); /* Start the Animation */
```

## 7.7.2 Animation Path

You can control the Path (curve) of an Animation. The simplest case is linear, meaning the current value between *start* and *end* is changed at the same rate (i.e. with fixed steps) over the duration of the Animation. A *Path* is a function which calculates the next value to set based on the current state of the Animation. There are a number of built-in *Paths* that can be used:

- `lv_anim_path_linear()`: linear Animation (default)
- `lv_anim_path_step()`: change in one step at the end
- `lv_anim_path_ease_in()`: slow at the beginning
- `lv_anim_path_ease_out()`: slow at the end
- `lv_anim_path_ease_in_out()`: slow at the beginning and end
- `lv_anim_path_overshoot()`: overshoot the end value
- `lv_anim_path_bounce()`: bounce back a little from the end value (like hitting a wall)

Alternately, you can provide your own Path function.

`lv_anim_init(&my_anim)` sets the Path to `lv_anim_path_linear()` by default. If you want to use a different Path (including a custom Path function you provide), you set it using `lv_anim_set_path_cb(&anim_template, path_cb)`.

If you provide your own custom Path function, its prototype is:

```
int32_t calculate_value(lv_anim_t * anim);
```

### 7.7.3 Speed vs Time

Normally, you set the Animation duration directly using `lv_anim_set_duration(&anim_template, duration_in_ms)`. But in some cases the `rate` is known but the duration is not known. Given an Animation's `start` and `end` values, `rate` here means the number of units of change per second, i.e. how quickly (units per second) the Animation's value needs to change between the `start` and `end` value. For such cases there is a utility function `lv_anim_speed_to_time()` you can use to compute the Animation's duration, so you can set it like this:

```
uint32_t change_per_sec = 20;
uint32_t duration_in_ms = lv_anim_speed_to_time(change_per_sec, 0, 100);
/* "duration_in_ms" will be 5000 */
lv_anim_set_duration(&anim_template, duration_in_ms);
```

### 7.7.4 Animating in Both Directions

Sometimes an Animation needs to play forward, and then play backwards, effectively reversing course, animating from the `end` value back to the `start` value again. To do this, pass a non-zero value to this function to set the duration for the reverse portion of the Animation: `lv_anim_set_reverse_duration(&anim_template, duration_in_ms)`.

Optionally, you can also introduce a delay between the forward and backward directions using `lv_anim_set_reverse_delay(&anim_template, delay_in_ms)`

### 7.7.5 Starting an Animation

After you have set up your `lv_anim_t` object, it is important to realize that what you have set up is a "template" for a live, running Animation that has not been created yet. When you call `lv_anim_start(&anim_template)` passing the `template` you have set up, it uses your template to dynamically allocate an internal object that is a *live, running* Animation. This function returns a pointer to that object.

```
static lv_anim_t anim_template;
static lv_anim_t * running_anim;

/* Set up template... */
lv_anim_init(&anim_template);
/* ...and other set-up functions above. */

/* Later... */
running_anim = lv_anim_start(&anim_template);
```

#### Nota

`lv_anim_start(&anim_template)` makes its own copy of the Animation template, so if you do not need it later, its contents do not need to be preserved after this call.

Once a *live running* Animation has been started, it runs until it has completed, or until it is deleted (see below), whichever comes first. An Animation has completed when:

- its "value" has reached the designated `end` value;
- if the Animation has a non-zero `reverse` duration value, then its value has run from the `end` value back to the `start` value again;
- if a non-zero repeat count has been set, it has repeated the Animation that number of times.

Once the *live, running* Animation reaches completion, it is automatically deleted from the list of running Animations. This does not impact your Animation template.

#### Nota

If `lv_anim_set_repeat_count(&anim_template, cnt)` has been called passing `LV_ANIM_REPEAT_INFINITE`, the animation never reaches a state of being "completed". In this case, it must be deleted to terminate the Animation.

## 7.7.6 Deleting Animations

You should delete an Animation using `lv_anim_delete(var, func)` if one of these two conditions exists:

- the object (variable) being animated is deleted (and it is not a Widget) or
- a running animation needs to be stopped before it is completed.

### Nota

If the object (variable) being deleted is a type of Widget, the housekeeping code involved in deleting it also deletes any running animations that are connected with it. So `lv_anim_delete(var, func)` only needs to be called if the object being deleted is *not* one of the Widgets.

If you kept a copy of the pointer returned by `lv_anim_start()` as `running_anim`, you can delete the running animation like this:

```
lv_anim_delete(running_anim->var, running_anim->exec_cb);
```

In the event that the Animation completes *after* you have determined it needs to be deleted, and before the call to `lv_anim_delete()` is made, it does no harm to call it a second time --- no damage will occur.

This function returns a Boolean value indicating whether any *live, running* Animations were deleted.

## 7.7.7 Pausing Animations

If you kept a copy of the pointer returned by `lv_anim_start()`, you can pause the running animation using `lv_anim_pause(animation)` and then resume it using `lv_anim_resume(animation)`.

`lv_anim_pause_for(animation, milliseconds)` is also available if you wish for the animation to resume automatically after.

## 7.7.8 Timeline

You can create a series of related animations that are linked together using an Animation Timeline. A Timeline is a collection of multiple Animations which makes it easy to create complex composite Animations. To create and use an Animation Timeline:

- Create an Animation template but do not call `lv_anim_start()` on it.
- Create an Animation Timeline object by calling `lv_anim_timeline_create()`.
- Add Animation templates to the Timeline by calling `lv_anim_timeline_add(timeline, start_time, &anim_template)`. `start_time` is the start time of the Animation on the Timeline. Note that `start_time` will override any value given to `lv_anim_set_delay(&anim_template, delay)`.
- Call `lv_anim_timeline_start(timeline)` to start the Animation Timeline.

### Nota

`lv_anim_timeline_add(timeline, start_time, &anim_template)` makes its own copy of the contents of the Animation template, so if you do not need it later, its contents do not need to be preserved after this call.

It supports forward and reverse play of the entire Animation group, using `lv_anim_timeline_set_reverse(timeline, reverse)`. Note that if you want to play in reverse from the end of the Timeline, you need to call `lv_anim_timeline_set_progress(timeline, LV_ANIM_TIMELINE_PROGRESS_MAX)` after adding all Animations and before telling it to start playing.

Call `lv_anim_timeline_pause(timeline)` to pause the Animation Timeline. Note: this does not preserve its state. The only way to start it again is to call `lv_anim_timeline_start(timeline)`, which starts the Timeline from the beginning or at the point set by `lv_anim_timeline_set_progress(timeline, progress)`.

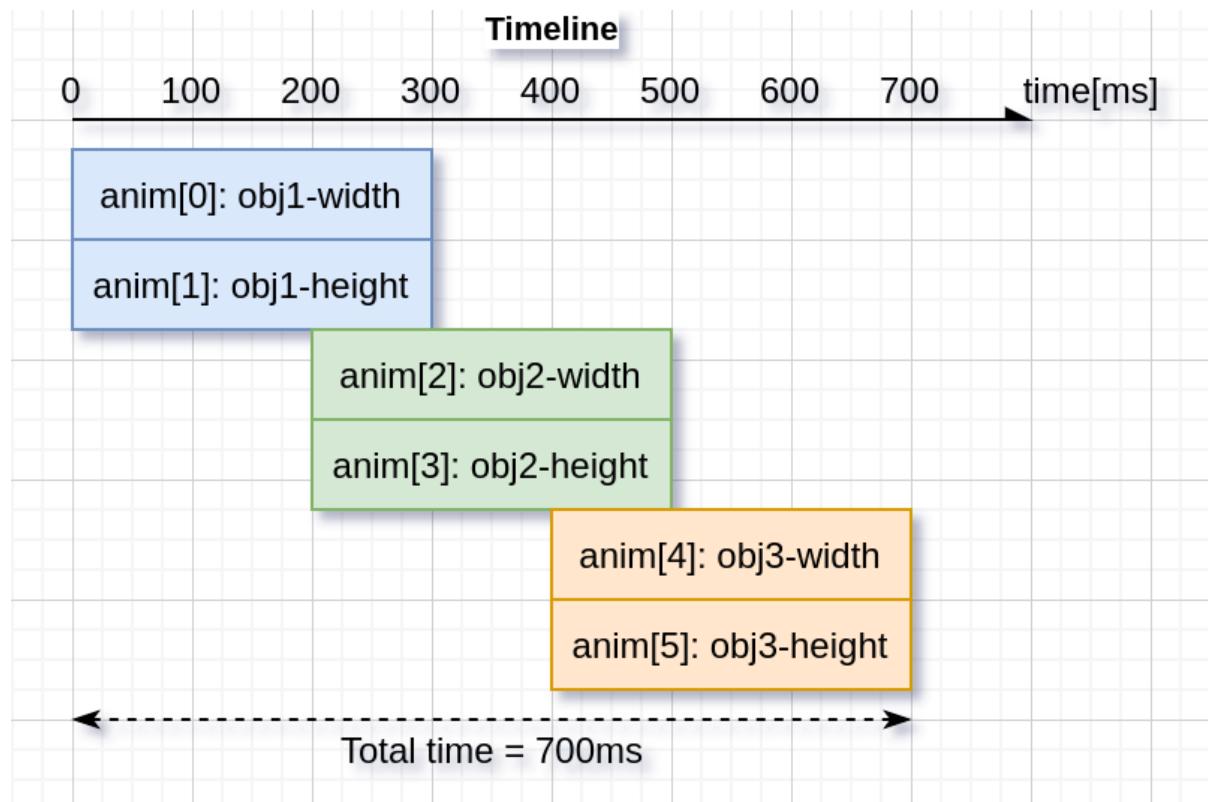
Call `lv_anim_timeline_set_progress(timeline, progress)` function to set the state of the Animation Timeline according to the `progress` value. `progress` is a value between 0 and 32767 (`LV_ANIM_TIMELINE_PROGRESS_MAX`) to indicate the proportion of the Timeline that has "played". Example: a `progress` value of `LV_ANIM_TIMELINE_PROGRESS_MAX / 2` would set the Timeline play to its half-way point.

Call `lv_anim_timeline_get_playtime(timeline)` function to get the total duration (in milliseconds) of the entire Animation Timeline.

Call `lv_anim_timeline_get_reverse(timeline)` function to get whether the Animation Timeline is also played in reverse after its forward play completes.

Call `lv_anim_timeline_delete(timeline)` function to delete the Animation Timeline. **Note:** If you need to delete a Widget during Animation, be sure to delete the Animation Timeline before deleting the Widget. Otherwise, the program may crash or behave abnormally.

If a base object is set with `lv_anim_timeline_set_base_object(timeline, obj)`, `var` in the added animations is assumed to be a widget name (or path) string. The actual widgets are retrieved by `lv_obj_get_child_by_name` before calling the `exec_cb` of the animation. That is, the `exec_cb` gets a pointer to the widget, and not the name/path.



## 7.7.9 Esempi

### Avviare animazione su un evento

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) var, v);
}

static void sw_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *) lv_event_get_user_data(e);

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), 100);
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_overshoot);
        lv_anim_start(&a);
    }
    else {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, -lv_obj_get_width(label));
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_ease_in);
        lv_anim_start(&a);
    }
}

/**
 * Start animation on an event
 */
void lv_example_anim_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello animations!");
    lv_obj_set_pos(label, 100, 10);

    lv_obj_t * sw = lv_switch_create(lv_screen_active());
    lv_obj_center(sw);
    lv_obj_add_state(sw, LV_STATE_CHECKED);
    lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_VALUE_CHANGED, label);
}
#endif
```

### Riproduzione di animazione

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) var, v);
}

static void anim_size_cb(void * var, int32_t v)
{
    lv_obj_set_size((lv_obj_t *) var, v, v);
}

/**
 * Create a playback animation
 */
void lv_example_anim_2(void)
{
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_remove_flag(obj, LV_OBJ_FLAG_SCROLLABLE);
    lv_obj_set_style_bg_color(obj, lv_palette_main(LV_PALETTE_RED), 0);
    lv_obj_set_style_radius(obj, LV_RADIUS_CIRCLE, 0);

    lv_obj_align(obj, LV_ALIGN_LEFT_MID, 10, 0);

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_var(&a, obj);
    lv_anim_set_values(&a, 10, 50);
    lv_anim_set_duration(&a, 1000);
    lv_anim_set_reverse_delay(&a, 100);
    lv_anim_set_reverse_duration(&a, 300);
    lv_anim_set_repeat_delay(&a, 500);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_set_path_cb(&a, lv_anim_path_ease_in_out);

    lv_anim_set_exec_cb(&a, anim_size_cb);
    lv_anim_start(&a);
    lv_anim_set_exec_cb(&a, anim_x_cb);
    lv_anim_set_values(&a, 10, 240);
    lv_anim_start(&a);
}

#endif
```

## Animazione di cubica di Bézier

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SLIDER && LV_USE_CHART && LV_USE_BUTTON && LV_USE_GRID

/***
 * the example show the use of cubic-bezier3 in animation.
 * the control point P1,P2 of cubic-bezier3 can be adjusted by slider.
 * and the chart shows the cubic-bezier3 in real time. you can click
 * run button see animation in current point of cubic-bezier3.
 */

#define CHART_POINTS_NUM 256

struct {
    lv_obj_t * anim_obj;
    lv_obj_t * chart;
    lv_chart_series_t * ser1;
    lv_obj_t * p1_slider;
    lv_obj_t * p1_label;
    lv_obj_t * p2_slider;
    lv_obj_t * p2_label;
    lv_obj_t * run_btn;
    int32_t p1;
    int32_t p2;
    lv_anim_t a;
} ginfo;

static int32_t anim_path_bezier3_cb(const lv_anim_t * a);
static void refer_chart_cubic_bezier(void);
static void run_button_event_handler(lv_event_t * e);
static void slider_event_cb(lv_event_t * e);
static void page_obj_init(lv_obj_t * par);
static void anim_x_cb(void * var, int32_t v);

/***
 * create an animation
 */
void lv_example_anim_3(void)
{
    static int32_t col_dsc[] = {LV_GRID_FR(1), 200, LV_GRID_FR(1), LV_GRID_TEMPLATE_LAST};
    static int32_t row_dsc[] = {30, 10, 10, LV_GRID_FR(1), LV_GRID_TEMPLATE_LAST};

    /*Create a container with grid*/
    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_style_pad_all(cont, 2, LV_PART_MAIN);
    lv_obj_set_style_pad_column(cont, 10, LV_PART_MAIN);
    lv_obj_set_style_pad_row(cont, 10, LV_PART_MAIN);
    lv_obj_set_grid_dsc_array(cont, col_dsc, row_dsc);
    lv_obj_set_size(cont, 320, 240);
    lv_obj_center(cont);

    page_obj_init(cont);

    lv_anim_init(&ginfo.a);
    lv_anim_set_var(&ginfo.a, ginfo.anim_obj);
    int32_t end = lv_obj_get_style_width(cont, LV_PART_MAIN) -
        lv_obj_get_style_width(ginfo.anim_obj, LV_PART_MAIN) - 10;
    lv_anim_set_values(&ginfo.a, 5, end);
    lv_anim_set_duration(&ginfo.a, 2000);
    lv_anim_set_exec_cb(&ginfo.a, anim_x_cb);
    lv_anim_set_path_cb(&ginfo.a, anim_path_bezier3_cb);

    refer_chart_cubic_bezier();
}

static int32_t anim_path_bezier3_cb(const lv_anim_t * a)
{
    int32_t t = lv_map(a->act_time, 0, a->duration, 0, 1024);
    int32_t step = lv_bezier3(t, 0, ginfo.p1, ginfo.p2, 1024);
    int32_t new_value;
    new_value = step * (a->end_value - a->start_value);
    new_value = new_value >> 10;
    new_value += a->start_value;
    return new_value;
}

static void refer_chart_cubic_bezier(void)
{
    for(uint16_t i = 0; i <= CHART_POINTS_NUM; i++) {
        int32_t t = i * (1024 / CHART_POINTS_NUM);
        int32_t step = lv_bezier3(t, 0, ginfo.p1, ginfo.p2, 1024);
        lv_chart_set_series_value_by_id2(ginfo.chart, ginfo.ser1, i, t, step);
    }
    lv_chart_refresh(ginfo.chart);
}

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_style_translate_x((lv_obj_t *)var, v, LV_PART_MAIN);
}

static void run_button_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    if(code == LV_EVENT_CLICKED) {
        lv_anim_start(&ginfo.a);
    }
}

static void slider_event_cb(lv_event_t * e)
{
    char buf[16];
    lv_obj_t * label;
    lv_obj_t * slider = lv_event_get_target_obj(e);

    if(slider == ginfo.p1_slider) {
        label = ginfo.p1_label;
        ginfo.p1 = lv_slider_get_value(slider);
        lv_snprintf(buf, sizeof(buf), "p1:%d", ginfo.p1);
    }
    else {
        label = ginfo.p2_label;
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

        ginfo.p2 = lv_slider_get_value(slider);
        lv_snprintf(buf, sizeof(buf), "p2:%d", ginfo.p2);
    }

    lv_label_set_text(label, buf);
    refer_chart_cubic();
}

static void page_obj_init(lv_obj_t * par)
{
    ginfo.anim_obj = lv_obj_create(par);
    lv_obj_set_size(ginfo.anim_obj, 30, 30);
    lv_obj_set_align(ginfo.anim_obj, LV_ALIGN_TOP_LEFT);
    lv_obj_remove_flag(ginfo.anim_obj, LV_OBJ_FLAG_SCROLLABLE);
    lv_obj_set_style_bg_color(ginfo.anim_obj, lv_palette_main(LV_PALETTE_RED), LV_PART_MAIN);
    lv_obj_set_grid_cell(ginfo.anim_obj, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 0, 1);

    ginfo.p1_label = lv_label_create(par);
    ginfo.p2_label = lv_label_create(par);
    lv_label_set_text(ginfo.p1_label, "p1:0");
    lv_label_set_text(ginfo.p2_label, "p2:0");
    lv_obj_set_grid_cell(ginfo.p1_label, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 1, 1);
    lv_obj_set_grid_cell(ginfo.p2_label, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_START, 2, 1);

    ginfo.p1_slider = lv_slider_create(par);
    ginfo.p2_slider = lv_slider_create(par);
    lv_slider_set_range(ginfo.p1_slider, 0, 1024);
    lv_slider_set_range(ginfo.p2_slider, 0, 1024);
    lv_obj_set_style_pad_all(ginfo.p1_slider, 2, LV_PART_KNOB);
    lv_obj_set_style_pad_all(ginfo.p2_slider, 2, LV_PART_KNOB);
    lv_obj_add_event_cb(ginfo.p1_slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_add_event_cb(ginfo.p2_slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_grid_cell(ginfo.p1_slider, LV_GRID_ALIGN_STRETCH, 1, 1, LV_GRID_ALIGN_START, 1, 1);
    lv_obj_set_grid_cell(ginfo.p2_slider, LV_GRID_ALIGN_STRETCH, 1, 1, LV_GRID_ALIGN_START, 2, 1);

    ginfo.run_btn = lv_button_create(par);
    lv_obj_add_event_cb(ginfo.run_btn, run_button_event_handler, LV_EVENT_CLICKED, NULL);
    lv_obj_t * btn_label = lv_label_create(ginfo.run_btn);
    lv_label_set_text(btn_label, LV_SYMBOL_PLAY);
    lv_obj_center(btn_label);
    lv_obj_set_grid_cell(ginfo.run_btn, LV_GRID_ALIGN_STRETCH, 2, 1, LV_GRID_ALIGN_STRETCH, 1, 2);

    ginfo.chart = lv_chart_create(par);
    lv_obj_set_style_pad_all(ginfo.chart, 0, LV_PART_MAIN);
    lv_obj_set_style_size(ginfo.chart, 0, 0, LV_PART_INDICATOR);
    lv_chart_set_type(ginfo.chart, LV_CHART_TYPE_SCATTER);
    ginfo.ser1 = lv_chart_add_series(ginfo.chart, lv_palette_main(LV_PALETTE_RED), LV_CHART_AXIS_PRIMARY_Y);
    lv_chart_set_axis_range(ginfo.chart, LV_CHART_AXIS_PRIMARY_Y, 0, 1024);
    lv_chart_set_axis_range(ginfo.chart, LV_CHART_AXIS_PRIMARY_X, 0, 1024);
    lv_chart_set_point_count(ginfo.chart, CHART_POINTS_NUM);
    lv_obj_set_grid_cell(ginfo.chart, LV_GRID_ALIGN_STRETCH, 0, 3, LV_GRID_ALIGN_STRETCH, 3, 1);
}

#endif

```

## Pausa animazione

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_SWITCH

static void anim_x_cb(void * var, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) var, v);
}

static void timer_cb(lv_timer_t * timer)
{
    lv_anim_t * anim = (lv_anim_t *) lv_timer_get_user_data(timer);
    lv_anim_pause_for(anim, 1000);
    lv_timer_delete(timer);
}
static void sw_event_cb(lv_event_t * e)
{
    lv_obj_t * sw = lv_event_get_target_obj(e);
    lv_obj_t * label = (lv_obj_t *) lv_event_get_user_data(e);

    if(lv_obj_has_state(sw, LV_STATE_CHECKED)) {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), 100);
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_overshoot);
        lv_timer_create(timer_cb, 200, lv_anim_start(&a));
    }
    else {
        lv_anim_t a;
        lv_anim_init(&a);
        lv_anim_set_var(&a, label);
        lv_anim_set_values(&a, lv_obj_get_x(label), -lv_obj_get_width(label));
        lv_anim_set_duration(&a, 500);
        lv_anim_set_exec_cb(&a, anim_x_cb);
        lv_anim_set_path_cb(&a, lv_anim_path_ease_in);
        lv_timer_create(timer_cb, 200, lv_anim_start(&a));
    }
}

/**
 * Start animation on an event
 */
void lv_example_anim_4(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Hello animations!");
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_pos(label, 100, 10);
lv_obj_t * sw = lv_switch_create(lv_screen_active());
lv_obj_center(sw);
lv_obj_add_state(sw, LV_STATE_CHECKED);
lv_obj_add_event_cb(sw, sw_event_cb, LV_EVENT_VALUE_CHANGED, label);
}

#endif

```

## Timeline animazione

```

#include "../lv_examples.h"
#if LV_USE_FLEX && LV_BUILD_EXAMPLES

static const int32_t obj_width = 90;
static const int32_t obj_height = 70;

static void set_width(lv_anim_t * var, int32_t v)
{
    lv_obj_set_width((lv_obj_t *) var->var, v);
}

static void set_height(lv_anim_t * var, int32_t v)
{
    lv_obj_set_height((lv_obj_t *) var->var, v);
}

static void set_slider_value(lv_anim_t * var, int32_t v)
{
    lv_slider_set_value((lv_obj_t *) var->var, v, LV_ANIM_OFF);
}

static void btn_start_event_handler(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_current_target_obj(e);
    lv_anim_timeline_t * anim_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);

    bool reverse = lv_obj_has_state(btn, LV_STATE_CHECKED);
    lv_anim_timeline_set_reverse(anim_timeline, reverse);
    lv_anim_timeline_start(anim_timeline);
}

static void btn_pause_event_handler(lv_event_t * e)
{
    lv_anim_timeline_t * anim_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);
    lv_anim_timeline_pause(anim_timeline);
}

static void slider_prg_event_handler(lv_event_t * e)
{
    lv_obj_t * slider = lv_event_get_current_target_obj(e);
    lv_anim_timeline_t * anim_timeline = (lv_anim_timeline_t *) lv_event_get_user_data(e);
    int32_t progress = lv_slider_get_value(slider);
    lv_anim_timeline_set_progress(anim_timeline, (uint16_t) progress);
}

/***
 * Create an animation timeline
 */
void lv_example_anim_timeline_1(void)
{
    /* Create anim timeline */
    lv_anim_timeline_t * anim_timeline = lv_anim_timeline_create();

    lv_obj_t * par = lv_screen_active();
    lv_obj_set.flex_flow(par, LV_FLEX_FLOW_ROW);
    lv_obj_set.flex_align(par, LV_FLEX_ALIGN_SPACE_AROUND, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);

    /* create btn_start */
    lv_obj_t * btn_start = lv_button_create(par);
    lv_obj_add_event_cb(btn_start, btn_start_event_handler, LV_EVENT_VALUE_CHANGED, anim_timeline);
    lv_obj_add_flag(btn_start, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(btn_start, LV_ALIGN_TOP_MID, -100, 20);

    lv_obj_t * label_start = lv_label_create(btn_start);
    lv_label_set_text(label_start, "Start");
    lv_obj_center(label_start);

    /* create btn_pause */
    lv_obj_t * btn_pause = lv_button_create(par);
    lv_obj_add_event_cb(btn_pause, btn_pause_event_handler, LV_EVENT_CLICKED, anim_timeline);
    lv_obj_add_flag(btn_pause, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(btn_pause, LV_ALIGN_TOP_MID, 100, 20);

    lv_obj_t * label_pause = lv_label_create(btn_pause);
    lv_label_set_text(label_pause, "Pause");
    lv_obj_center(label_pause);

    /* create slider_prg */
    lv_obj_t * slider_prg = lv_slider_create(par);
    lv_obj_add_event_cb(slider_prg, slider_prg_event_handler, LV_EVENT_VALUE_CHANGED, anim_timeline);
    lv_obj_add_flag(slider_prg, LV_OBJ_FLAG_IGNORE_LAYOUT);
    lv_obj_align(slider_prg, LV_ALIGN_BOTTOM_MID, 0, -20);
    lv_slider_set_range(slider_prg, 0, LV_ANIM_TIMELINE_PROGRESS_MAX);

    /* create 3 objects */
    lv_obj_t * obj1 = lv_obj_create(par);
    lv_obj_set_size(obj1, obj_width, obj_height);
    lv_obj_set_scrollbar_mode(obj1, LV_SCROLLBAR_MODE_OFF);

    lv_obj_t * obj2 = lv_obj_create(par);
    lv_obj_set_size(obj2, obj_width, obj_height);
    lv_obj_set_scrollbar_mode(obj2, LV_SCROLLBAR_MODE_OFF);

    lv_obj_t * obj3 = lv_obj_create(par);
    lv_obj_set_size(obj3, obj_width, obj_height);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_scrollbar_mode(obj3, LV_SCROLLBAR_MODE_OFF);

/* anim-slider */
lv_anim_t a_slider;
lv_anim_init(&a_slider);
lv_anim_set_var(&a_slider, slider_prg);
lv_anim_set_values(&a_slider, 0, LV_ANIM_TIMELINE_PROGRESS_MAX);
lv_anim_set_custom_exec_cb(&a_slider, set_slider_value);
lv_anim_set_path_cb(&a_slider, lv_anim_path_linear);
lv_anim_set_duration(&a_slider, 700);

/* anim-obj1 */
lv_anim_t a1;
lv_anim_init(&a1);
lv_anim_set_var(&a1, obj1);
lv_anim_set_values(&a1, 0, obj_width);
lv_anim_set_custom_exec_cb(&a1, set_width);
lv_anim_set_path_cb(&a1, lv_anim_path_overshoot);
lv_anim_set_duration(&a1, 300);

lv_anim_t a2;
lv_anim_init(&a2);
lv_anim_set_var(&a2, obj1);
lv_anim_set_values(&a2, 0, obj_height);
lv_anim_set_custom_exec_cb(&a2, set_height);
lv_anim_set_path_cb(&a2, lv_anim_path_ease_out);
lv_anim_set_duration(&a2, 300);

/* anim-obj2 */
lv_anim_t a3;
lv_anim_init(&a3);
lv_anim_set_var(&a3, obj2);
lv_anim_set_values(&a3, 0, obj_width);
lv_anim_set_custom_exec_cb(&a3, set_width);
lv_anim_set_path_cb(&a3, lv_anim_path_overshoot);
lv_anim_set_duration(&a3, 300);

lv_anim_t a4;
lv_anim_init(&a4);
lv_anim_set_var(&a4, obj2);
lv_anim_set_values(&a4, 0, obj_height);
lv_anim_set_custom_exec_cb(&a4, set_height);
lv_anim_set_path_cb(&a4, lv_anim_path_ease_out);
lv_anim_set_duration(&a4, 300);

/* anim-obj3 */
lv_anim_t a5;
lv_anim_init(&a5);
lv_anim_set_var(&a5, obj3);
lv_anim_set_values(&a5, 0, obj_width);
lv_anim_set_custom_exec_cb(&a5, set_width);
lv_anim_set_path_cb(&a5, lv_anim_path_overshoot);
lv_anim_set_duration(&a5, 300);

lv_anim_t a6;
lv_anim_init(&a6);
lv_anim_set_var(&a6, obj3);
lv_anim_set_values(&a6, 0, obj_height);
lv_anim_set_custom_exec_cb(&a6, set_height);
lv_anim_set_path_cb(&a6, lv_anim_path_ease_out);
lv_anim_set_duration(&a6, 300);

/* add animations to timeline */
lv_anim_timeline_add(anim_timeline, 0, &a_slider);
lv_anim_timeline_add(anim_timeline, 0, &a1);
lv_anim_timeline_add(anim_timeline, 0, &a2);
lv_anim_timeline_add(anim_timeline, 200, &a3);
lv_anim_timeline_add(anim_timeline, 200, &a4);
lv_anim_timeline_add(anim_timeline, 400, &a5);
lv_anim_timeline_add(anim_timeline, 400, &a6);

lv_anim_timeline_set_progress(anim_timeline, LV_ANIM_TIMELINE_PROGRESS_MAX);
}

#endif

```

## 7.7.10 API

## 7.8 File System (lv\_fs\_drv)

LVGL has a "File system" abstraction module that enables you to attach any type of file system. A file system is identified by an assigned identifier letter. For example, if an SD card is associated with the letter 'S', a file can be reached using "S:/path/to/file.txt". See details under [Identifier Letters](#).

### Nota

If you want to skip the drive-letter prefix in Unix-like paths, you can use the `LV_FS_DEFAULT_DRIVE LETTER` config parameter.

## 7.8.1 Ready-to-Use Drivers

LVGL contains prepared drivers for the API of POSIX, standard C, Windows, and **FATFS**. Learn more [here](#).

## 7.8.2 Identifier Letters

As mentioned above, a file system is identified by an assigned identifier letter. This identifier is merely a way for the LVGL File System abstraction logic to look up the appropriate registered file-system driver for a given path.

### How it Works:

You register a driver for your file system and assign it an identifier letter. This letter must be unique among all registered file-system drivers, and in the range [A-Z] or the character '/'. See [Adding a Driver](#) for how this is done.

Later, when using paths to files on your file system, you prefix the path with that identifier character plus a colon (':').

### Nota

Do not confuse this with a Windows or DOS drive letter.

### Example:

Let's use the letter 'Z' as the identifier character, and "path\_to\_file" as the path, then the path strings you pass to `lv_fs_...()` functions would look like this:

```
"Z:path_to_file"
^ ~~~~~
| |
| +-- This part gets passed to the OS-level file-system functions.
| +-- This part LVGL strips from path string, and uses it to find the appropriate
|     driver (i.e. set of functions) that apply to that file system.
```

Note also that the path can be a relative path or a "rooted path" (beginning with '/), though rooted paths are recommended since the driver does not yet provide a way to set the default directory.

### Examples for Unix-like file systems:

- "Z:/etc/images/splash.png"
- "Z:/etc/images/left\_button.png"
- "Z:/etc/images/right\_button.png"
- "Z:/home/users/me/wip/proposal.txt"

### Examples for Windows/DOS-like file systems:

- "Z:C:/Users/me/wip/proposal.txt"
- "Z:/Users/me/wip/proposal.txt" (if the default drive is known to be C:)
- "Z:C:/Users/Public/Documents/meeting\_notes.txt"
- "Z:D:/to\_print.docx"

Reminder: Note carefully that the prefixed "Z:" has nothing to do with the "C:" and "D:" Windows/DOS drive letters in 3 of the above examples, which are part of the path. "Z:" is used to look up the driver for that file system in the list of all file-system drivers registered with LVGL.

## 7.8.3 Adding a Driver

### Registering a driver

To add a driver, a `lv_fs_drv_t` object needs to be initialized and registered in a way similar to the code below. The `lv_fs_drv_t` variable needs to be static, global or dynamically allocated and not a local variable, since its contents need to remain valid as long as the driver is in use.

```

static lv_fs_drv_t drv;           /* Needs to be static or global */
lv_fs_drv_init(&drv);          /* Basic initialization */

drv.letter = 'S';               /* An uppercase letter to identify the drive */
drv.cache_size = my_cache_size;  /* Cache size for reading in bytes. 0 to not cache. */

drv.ready_cb = my_ready_cb;     /* Callback to tell if the drive is ready to use */
drv.open_cb = my_open_cb;        /* Callback to open a file */
drv.close_cb = my_close_cb;     /* Callback to close a file */
drv.read_cb = my_read_cb;       /* Callback to read a file */
drv.write_cb = my_write_cb;     /* Callback to write a file */
drv.seek_cb = my_seek_cb;       /* Callback to seek in a file (Move cursor) */
drv.tell_cb = my_tell_cb;       /* Callback to tell the cursor position */

drv.dir_open_cb = my_dir_open_cb; /* Callback to open directory to read its content */
drv.dir_read_cb = my_dir_read_cb; /* Callback to read a directory's content */
drv.dir_close_cb = my_dir_close_cb; /* Callback to close a directory */

drv.user_data = my_user_data;   /* Any custom data if required */

lv_fs_drv_register(&drv);      /* Finally register the drive */

```

Any of the callbacks can be **NULL** to indicate that operation is not supported.

## Implementing the callbacks

### Open callback

The prototype of **open\_cb** looks like this:

```
void * (*open_cb)(lv_fs_drv_t *drv, const char *path, lv_fs_mode_t mode);
```

**path** is the path after the drive letter (e.g. "S:path/to/file.txt" -> "path/to/file.txt"). **mode** can be **LV\_FS\_MODE\_WR** or **LV\_FS\_MODE\_RD** to open for writes or reads.

The return value is a pointer to a *file object* that describes the opened file or **NULL** if there were any issues (e.g. the file wasn't found). The returned file object will be passed to other file system related callbacks. (Vedere di seguito).

### Other callbacks

The other callbacks are quite similar. For example **write\_cb** looks like this:

```
lv_fs_res_t (*write_cb)(lv_fs_drv_t *drv, void *file_p, const void *buf, uint32_t btw, uint32_t *bw);
```

For **file\_p**, LVGL passes the return value of **open\_cb**, **buf** is the data to write, **btw** is the number of "bytes to write", **bw** is the number of "bytes written" (written to during the function call).

For a list of prototypes for these callbacks see **lv\_fs\_template.c**. This file also provides a template for new file-system drivers you can use if the one you need is not already provided.

## Drivers that come with LVGL

As of this writing, the list of already-available file-system drivers can be enabled by setting one or more of the following macros to a non-zero value in **lv\_conf.h**. The drivers are as implied by the macro names.

If you use more than one, each associated identifier letter you use must be unique.

- **LV\_USE\_FS\_FATFS**
- **LV\_USE\_FS\_STUDIO**
- **LV\_USE\_FS\_POSIX**
- **LV\_USE\_FS\_WIN32**
- **LV\_USE\_FS\_MEMFS**
- **LV\_USE\_FS\_LITTLEFS**
- **LV\_USE\_FS\_ARDUINO\_ESP\_LITTLEFS**
- **LV\_USE\_FS\_ARDUINO\_SD**

## 7.8.4 Limiting Directory Access

If you are using one of the following file-system drivers:

- *LV\_USE\_FS\_STUDIO*
- *LV\_USE\_FS\_POSIX*
- *LV\_USE\_FS\_WIN32*

you will have a *LV\_FS\_xxx\_PATH* macro available to you in *lv\_conf.h* that you can use to provide a path that gets dynamically prefixed to the *path\_to\_file* portion of the path strings provided to *lv\_fs\_...()* functions when files and directories are opened. This can be useful to limit directory access (e.g. when a portion of a path can be typed by an end user), or simply to reduce the length of the path strings provided to *lv\_fs\_...()* functions.

Do this by filling in the full path to the directory you wish his access to be limited to in the applicable *LV\_FS\_xxx\_PATH* macro in *lv\_conf.h*. Do not prefix the path with the driver-identifier letter, and do append a directory separator character at the end.

### Examples for Unix-like file systems:

```
#define LV_FS_WIN32_PATH "/home/users/me/"
```

### Examples for Windows/DOS-like file systems:

```
#define LV_FS_WIN32_PATH "C:/Users/me/"
```

Then in both cases, path strings passed to *lv\_fs\_...()* functions in the application get reduced to:

- "Z:wip/proposal.txt"

## 7.8.5 Usage Example

The example below shows how to read from a file:

```
lv_fs_file_t f;
lv_fs_res_t res;
res = lv_fs_open(&f, "S:folder/file.txt", LV_FS_MODE_RD);
if(res != LV_FS_RES_OK) my_error_handling();

uint32_t read_num;
uint8_t buf[8];
res = lv_fs_read(&f, buf, 8, &read_num);
if(res != LV_FS_RES_OK || read_num != 8) my_error_handling();

lv_fs_close(&f);
```

The mode in *lv\_fs\_open()* can be *LV\_FS\_MODE\_WR* to open for writes only, *LV\_FS\_MODE\_RD* for reads only, or *LV\_FS\_MODE\_RD | LV\_FS\_MODE\_WR* for both.

This example shows how to read a directory's content. It's up to the driver how to mark directories in the result but it can be a good practice to insert a '/' in front of each directory name.

```
lv_fs_dir_t dir;
lv_fs_res_t res;
res = lv_fs_dir_open(&dir, "S:/folder");
if(res != LV_FS_RES_OK) my_error_handling();

char fn[256];
while(1) {
    res = lv_fs_dir_read(&dir, fn, sizeof(fn));
    if(res != LV_FS_RES_OK) {
        my_error_handling();
        break;
    }

    /* fn is empty if there are no more files to read. */
    if(strlen(fn) == 0) {
        break;
    }

    printf("%s\n", fn);
}

lv_fs_dir_close(&dir);
```

## 7.8.6 Use Drives for Images

*Image* Widgets can be opened from files as well (besides variables stored in the compiled program).

To use files in Image Widgets the following callbacks are required:

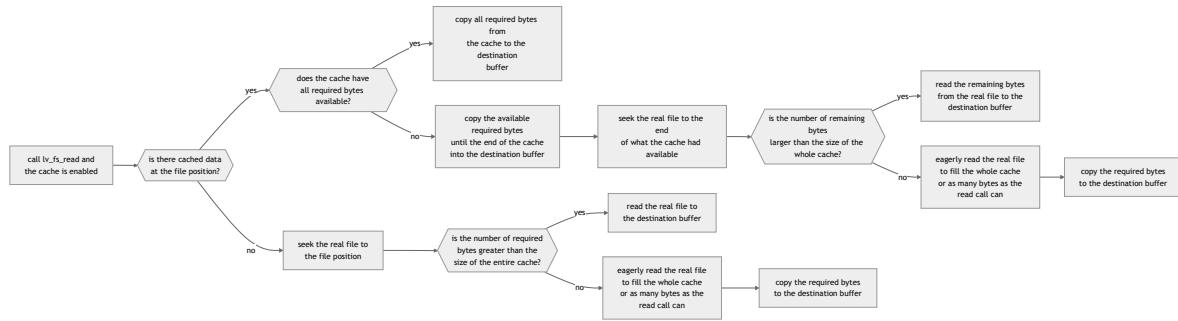
- open
- close
- read
- seek
- tell

## 7.8.7 Optional File Buffering/Caching

Files will buffer their reads if the corresponding `LV_FS_*_CACHE_SIZE` config option is set to a value greater than zero. Each open file will buffer up to that many bytes to reduce the number of FS driver calls.

Generally speaking, file buffering can be optimized for different kinds of access patterns. The one implemented here is optimal for reading large files in chunks, which is what the image decoder does. It has the potential to call the driver's `read` fewer times than `lv_fs_read` is called. In the best case where the cache size is  $\geq$  the size of the file, `read` will only be called once. This strategy is good for linear reading of large files but less helpful for short random reads across a file bigger than the buffer since data will be buffered that will be discarded after the next seek and read. The cache should be sufficiently large or disabled in that case. Another case where the cache should be disabled is if the file contents are expected to change by an external factor like with special OS files.

The implementation is documented below. Note that the FS functions make calls to other driver FS functions when the cache is enabled. i.e., `lv_fs_read` may call the driver's `seek` so the driver needs to implement more callbacks when the cache is enabled.

**lv\_fs\_read** (behavior when cache is enabled)

(Use mouse wheel to zoom in, and drag image to pan it.)

**lv\_fs\_write** (behavior when cache is enabled)

The part of the cache that coincides with the written content will be updated to reflect the written content.

**lv\_fs\_seek** (behavior when cache is enabled)

The driver's `seek` will not actually be called unless the whence is `LV_FS_SEEK_END`, in which case `seek` and `tell` will be called to determine where the end of the file is.

**lv\_fs\_tell** (behavior when cache is enabled)

The driver's `tell` will not actually be called.

## 7.8.8 API

`lv_fsdrv.h`

`lv_fs.h`

`lv_fs_private.h`

## 7.9 Drawing

### 7.9.1 Draw Pipeline

#### What is Drawing?

Drawing (also known as *rendering*) is writing pixel colors into a buffer where they will be delivered to a display panel as pixels. Sometimes this is done by copying colors from places like background- and foreground-color properties. Other times it involves computing those colors before they are written (e.g. combining them with other colors when an object higher on the Z axis has partial opacity).

The following sections cover the LVGL drawing logic and how to use it and optionally tune it to fit your particular project (e.g. if you have a GPU or other resources that you would like to get involved).

#### Draw-Pipeline Overview

On modern computing hardware meant to be used with larger display panels, there are sometimes options for different ways drawing can be accomplished. For example, some MCUs come with hardware that is very good (and fast) at certain types of drawing tasks. Alternatively, you might have access to a drawing library that performs certain types of drawing tasks with great efficiency. To make it possible to utilize such facilities in the most efficient fashion, LVGL v9 and onwards implements a *Drawing Pipeline*, like an assembly line, where decisions are made as to which drawing tasks (*Draw Tasks*) are given to which "logic entities" (*Draw Units*) in order to be carried out.

This Pipeline is designed so that it is both flexible and extendable. You can use it to perform custom rendering with a GPU, or replace the parts of the built-in software rendering logic to any extent desired.

Using events, it's also possible to modify *Draw Tasks* or insert new ones as LVGL renders Widgets.

The following sections describe the basic terminology and concepts of rendering.

#### Draw Tasks

A "Draw Task" (`lv_draw_task_t`) is a package of information that is created at the beginning of the Drawing Pipeline when a request to draw is made. Functions such as `lv_draw_rect()` and `lv_draw_label()` create one or more Draw Tasks and pass them down the Drawing Pipeline. Each Draw Task carries all the information required to:

- compute which *Draw Unit* should receive this task, and
- give the Draw Unit all the information required to accomplish the drawing task.

A Draw Task carries the following information:

<b>type</b>	defines the drawing algorithm involved (e.g. line, fill, border, image, label, arc, triangle, etc.)
<b>area</b>	defines the rectangle in which drawing will occur
<b>transformation matrix</b>	if <code>LV_DRAW_TRANSFORM_USE_MATRIX</code> is configured to '1'
<b>state</b>	waiting, queued, in progress, completed
<b>drawing descriptor</b>	carries details of the drawing to be performed
<b>preferred Draw Unit ID</b>	identifier of the Draw Unit that should carry out this task
<b>preference score</b>	value describing the speed of the specified Draw Unit relative to software rendering (more on this below)
<b>next</b>	a link to the next Draw Task in the list.

Draw Tasks are collected in a list and periodically dispatched to Draw Units.

## Draw Units

A "Draw Unit" (based on `lv_draw_unit_t`) is any "logic entity" that can generate the output required by a *Draw Task*. This can be a CPU core, a GPU, a custom rendering library for specific Draw Tasks, or any entity capable of performing rendering.

For a reference implementation of a draw unit, see `lv_draw_sw.c`.

During LVGL's initialization (`lv_init()`), a list of Draw Units is created. If `LV_USE_DRAW_SW` is set to 1 in `lv_conf.h` (it is by default), the Software Drawing Unit enters itself at the head of that list. If your platform has other drawing units available, if they are configured to be used in `lv_conf.h`, they are added to this list during LVGL's initialization. If you are adding your own Draw Unit(s), you add each available drawing unit to that list by calling `lv_draw_create_unit(sizeof(your_draw_unit_t))`. With each call to that function, the newly-created draw unit is added to the head of that list, pushing already-existing draw units further back in the list, pushing the Draw Units created earlier farther back in the list. The order of this list (and thus the order in which *Draw Task Evaluation* is performed) is governed by the order in which each Draw Unit is created.

Building this list (and initializing the Draw Units) is normally handled automatically by configuring the available Draw Units in `lv_conf.h`, such as setting `LV_USE_DRAW_OPENGL` or `LV_USE_PXP` or `LV_USE_DRAW SDL` or `LV_USE_DRAW_VG_LITE` to 1. However, if you are introducing your own Draw Unit(s), you will need to create and initialize it (after `lv_init()`) as above. This will include several things, but setting its `evaluate_cb` and `dispatch_cb` callbacks (mentioned later) are two of them.

For an example of how draw-unit creation and initialization is done, see `lv_draw_sw_init()` in `lv_draw_sw.c` or the other draw units whose `init` functions are optionally called in `lv_init()`.

## Thread Priority

All draw units operate with a configurable thread priority which can be set using the `LV_DRAW_THREAD_PRIO` configuration option in `lv_conf.h`. This allows you to fine-tune the priority level across all drawing units, which is especially useful for systems with limited priority levels.

By default, draw units use `LV_THREAD_PRIO_HIGH` as their thread priority. This consistent approach ensures that all drawing units (software rendering, hardware accelerators like STM32 DMA2D, NXP VGLite, etc.) use the same priority level unless explicitly configured otherwise.

## Draw Task Evaluation

When each *Draw Task* is created, each existing Draw Unit is "consulted" as to its "appropriateness" for the task. It does this through an "evaluation callback" function pointer (a.k.a. `evaluate_cb`), which each Draw Unit sets (for itself) during its initialization. Normally, that evaluation:

- optionally examines the existing "preference score" for the task mentioned above,
- if it can accomplish that type of task (e.g. line drawing) faster than other Draw Units that have already reported, it writes its own "preference score" and "preferred Draw Unit ID" to the respective fields in the task.

In this way, by the time the evaluation sequence is complete, the task will contain the score and the ID of the Drawing Unit that will be used to perform that task when it is *dispatched*.

This logic, of course, can be overridden or redefined, depending on system design.

As a side effect, this also ensures that the same Draw Unit will be selected consistently, depending on the type (and nature) of the drawing task, avoiding any possible screen jitter in case more than one Draw Unit is capable of performing a given task type.

The sequence of the Draw Unit list (with the Software Draw Unit at the end) also ensures that the Software Draw Unit is the "buck-stops-here" Draw Unit: if no other Draw Unit reported it was better at a given drawing task, then the Software Draw Unit will handle it.

## Dispatching

While collecting Draw Tasks LVGL frequently dispatches the collected Draw Tasks to their assigned Draw Units. This is handled via the `dispatch_cb` of the Draw Units.

If a Draw Unit is busy with another Draw Task, it just returns. However, if it is available it can take a Draw Task.

`lv_draw_get_next_available_task(layer, previous_task, draw_unit_id)` is a useful helper function which is used by the `dispatch_cb` to get the next Draw Task it should act on. If it handled the task, it sets the Draw Task's `state` field to `LV_DRAW_TASK_STATE_FINISHED`. "Available" in this context means that has been queued and assigned to a given Draw Unit and is ready to be carried out. The ramifications of having multiple drawing threads are taken into account for this.

## Run-Time Object Hierarchy

All of the above have this relationship at run time:

- LVGL (global)
  - list of *Draw Units*
  - list of *Display(s)*
    - \* Layer(s): Each *Display object* has its own list of *Draw Layers*
      - Draw Tasks: Each Layer has its own list of *Draw Tasks*

## API

`lv_draw.h`

`lv_draw_label.h`

`lv_draw_private.h`

`lv_draw_rect.h`

`lv_draw_sw.h`

`lv_conf_internal.h`

## 7.9.2 Draw API

### Where to Use the Drawing API

In most cases you use LVGL's Drawing API through the API of Widgets: by creating buttons, labels, etc., and setting their styles, positions, and other properties. In these cases rendering (drawing) is handled internally and you don't see the *Drawing Pipeline* at all.

However there are three places where you can use LVGL's Drawing API directly.

1. **In the draw events of the Widgets:** There are event codes which are sent when the Widget needs to render itself:

- **`LV_EVENT_DRAW_MAIN_BEGIN`, `LV_EVENT_DRAW_MAIN`, `LV_EVENT_DRAW_MAIN_END`:** Triggered before, during, and after a Widget is drawn, respectively. Widget rendering typically occurs in `LV_EVENT_DRAW_MAIN`.
- **`LV_EVENT_DRAW_POST_BEGIN`, `LV_EVENT_DRAW_POST`, `LV_EVENT_DRAW_POST_END`:** Triggered before, during, and after all child Widgets are rendered, respectively. This can be useful for overlay-like drawings, such as scrollbars which should be rendered on top of any children.

These are relevant if a new Widget is implemented and it uses custom drawing.

2. **Modifying the created draw tasks:** The when a draw task is created for a Widget `LV_EVENT_DRAW_TASK_ADDED` is sent. In this event the created draw task can be modified or new draw tasks can be added. Typical use cases for this are modifying each bar of a bar chart, or cells of a table.

For performance reasons, this event is disabled by default. Enable it by setting the `LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS` flag on the Widget(s) you wish to emit this event.

3. **Draw to the Canvas Widget:** The drawing functions can be used directly to draw to a Canvas Widget. Doing so renders custom drawing to a buffer which can be used later as an image or a mask.

For more information see [Canvas \(`lv\_canvas`\)](#).

### Drawing API

The main components of LVGL's Drawing API are the `lv_draw_rect()`, `lv_draw_label()`, `lv_draw_image()`, and similar functions. When they are called `lv_draw_task_t` objects are created internally.

These functions have the following parameters:

- **Layer:** This is the target of the drawing. See details at [Draw Layers](#).
- **Draw Descriptor:** This is a large `struct` containing all the information about the drawing. See details of the draw descriptors at [Draw Descriptors](#).
- **Area** (in some cases): Specifies where to draw.

### Coordinate System

Some functions and draw descriptors require area or point parameters. These are always **absolute coordinates** on the display. For example, if the target layer is on a 800x480 display and the layer's area is (100,100) (200,200), to render a 10x10 object in the middle, the coordinates (145,145) (154,154) should be used (not (40,40) (49,49)).

Exception: for the Canvas Widget the layer is always assumed to be at the (0,0) coordinate, regardless of the Canvas Widget's position.

### API

[`lv\_draw\_arc.h`](#)

[`lv\_draw\_image.h`](#)

[`lv\_draw\_label.h`](#)

[`lv\_draw\_line.h`](#)

`lv_draw_mask.h`

`lv_draw_triangle.h`

### 7.9.3 Draw Layers

Not to be confused with a *Display's main 4 layers*, a *Draw Layer* is a buffer created during rendering, necessitated by certain style properties, so different sets of pixels are correctly combined. Factors requiring such layers are:

- partial opacity
- bit-mask being applied
- blend mode
- clipped corners (a bit-mask application)
- transformations
  - scale
  - skew
  - rotation

Later that layer will be merged to the screen or its parent layer at the correct point in the rendering sequence.

## Layer Types

### Simple Layer

The following style properties trigger the creation of a "Simple Layer":

- `opa_layered`
- `bitmap_mask_src`
- `blend_mode`

In this case, the Widget will be sliced into `LV_DRAW_SW_LAYER_SIMPLE_BUF_SIZE` sized chunks.

If there is no memory for a new chunk, LVGL will try allocating the layer after another chunk is rendered and freed.

### Transform Layer

The following style properties trigger the creation of a "Transform Layer":

- `transform_scale_x`
- `transform_scale_y`
- `transform_skew_x`
- `transform_skew_y`
- `transform_rotate`

Due to the nature of transformations, the Widget being transformed (and its children) must be rendered first, followed by the transformation step. This necessitates a temporary drawing area (layer), often larger than the Widget proper, to provide an area of adequate size for the transformation. LVGL tries to render as small area of the widget as possible, but due to the nature of transformations no slicing is possible in this case.

### Clip Corner

The `clip_corner` style property also causes LVGL to create a 2 layers with radius height for the top and bottom parts of the Widget.

## Getting the Current Layer

The first parameter of the `lv_draw_rect/label/etc` functions is a layer.

In most cases a layer is not created, but an existing layer is used to draw there.

The draw API can be used in these cases and the current layer can be used differently in each case:

1. **In draw events:** In `LV_EVENT_DRAW_MAIN/POST_BEGIN/...` events the Widget is being rendered to a layer of the display or another temporary layer created earlier during rendering. The current target layer can be retrieved using `lv_event_get_layer(e)`.

It also possible to create new layers in these events, but the previous layer is also required since it will be the parent layer in `lv_draw_layer()`.

2. **Modifying the created Draw Tasks:** In `LV_EVENT_DRAW_TASK_ADDED` the draw tasks created by `lv_draw_rect/label/etc` can be modified. It's not required to know the current layer to modify a draw task. However, if something new also needs to be drawn with `lv_draw_rect/label/etc` the current layer is also required.

The current layer can be read from the `base` draw descriptor. Ad esempio:

```
/* In LV_EVENT_DRAW_TASK_ADDED */
lv_draw_task_t * t = lv_event_get_draw_task(e);
lv_draw_base_dsc_t * draw_dsc = lv_draw_task_get_draw_dsc(t);

lv_layer_t * current_layer = draw_dsc.layer;
```

3. **Draw to the Canvas Widget:** The canvas itself doesn't store a layer, but one can be easily created and used like this:

```
/* Initialize a layer */
lv_layer_t layer;
lv_canvas_init_layer(canvas, &layer);

/* Draw something here */

/* Wait until the rendering is ready */
lv_canvas_finish_layer(canvas, &layer);
```

## Creating a New Layer

To create a new layer, use `lv_draw_layer_create()`:

```
lv_area_t layer_area = {10, 10, 80, 50}; /* Area of the new layer */
lv_layer_t * new_layer = lv_draw_layer_create(parent_layer, LV_COLOR_FORMAT_RGB565, &layer_area);
```

Once the layer is created, draw tasks can be added to it by using the *Draw API* and *Draw descriptors*. In most cases this means calling the `lv_draw_rect/label/etc` functions.

Finally, the layer must be rendered to its parent layer. Since a layer behaves similarly to an image, it can be rendered the same way as images:

```
lv_draw_image_dsc_t image_draw_dsc;
lv_draw_image_dsc_init(&image_draw_dsc);
image_draw_dsc.src = new_layer; /* Source image is the new layer. */
/* Draw new layer to parent layer. */
lv_draw_layer(parent_layer, &image_draw_dsc, &layer_area);
```

## Memory Considerations

### Layer Buffers

The buffer for a layer (where rendering occurs) is not allocated at creation. Instead, it is allocated by *Draw Units* when the first *Draw Task* is dispatched.

Layer buffers can be large, so ensure there is sufficient heap memory or increase `LV_MEM_SIZE` in `lv_conf.h`.

## Layer Type Memory Requirements

To save memory, LVGL can render certain types of layers in smaller chunks:

1. **Simple Layers:** Simple layers can be rendered in chunks. For example, with `opa_layered = 140`, only 10 lines of the layer can be rendered at a time, then the next 10 lines, and so on. This avoids allocating a large buffer for the entire layer. The buffer size for a chunk is set using `LV_DRAW_LAYER_SIMPLE_BUF_SIZE` in `lv_conf.h`.
2. **Transform Layers:** Transform Widgets cannot be rendered in chunks because transformations often affect pixels outside the given area. For such layers, LVGL allocates a buffer large enough to render the entire transformed area without limits.

## Memory Limit for Layers

The total memory available for layers at once is controlled by `LV_DRAW_LAYER_MAX_MEMORY` in `lv_conf.h`. If set to `0`, there is no limit.

## API

`lv_obj_event.h`

`lv_draw.h`

## 7.9.4 Draw Descriptors

### Panoramica

Each `Draw Task` type has its own draw descriptor type. For example, `lv_draw_label_dsc_t` is used for label drawing, `lv_draw_image_dsc_t` is used for image drawing.

When an `lv_draw_...` function is called, it creates a Draw Task, copies the draw descriptor into a `malloced` memory block, and frees it automatically when needed. Therefore, local draw descriptor variables can be safely used.

### Relation to Styles

In most cases, style properties map 1-to-1 to draw descriptor fields. Ad esempio:

- `label_dsc.color` corresponds to the `text_color` style property.
- `shadow_dsc.width`, `line_dsc.opa`, and `arc_dsc.width` map to `shadow_width`, `line_opa`, and `arc_width` in styles.

See [Proprietà di Stile](#) to see the list of style properties and what they mean.

### Base Draw Descriptor

In each draw descriptor there is a generic "base descriptor" with `lv_draw_dsc_base_t` type and with `base` in its name. For example `label_dsc.base`. This `struct` stores useful information about which Widget and part created the draw descriptor. See all the fields in `lv_draw_dsc_base_t`.

In an `LV_EVENT_DRAW_TASK_ADDED` event, the elements of the base draw descriptor are very useful to identify the Draw Task. Ad esempio:

```
/* In LV_EVENT_DRAW_TASK_ADDED */
lv_draw_task_t * t = lv_event_get_draw_task(e);
lv_draw_base_dsc_t * draw_dsc = lv_draw_task_get_draw_dsc(t);
draw_dsc.obj; /* The Widget for which the draw descriptor was created */
draw_dsc.part; /* The Widget part for which the draw descriptor was created
E.g. LV_PART_INDICATOR */
draw_dsc.id1; /* A Widget type specific ID (e.g. table row index).
See the docs of the given Widget. */
draw_dsc.id2;
draw_dsc.layer; /* The target layer.
Required when a new Draw Tasks are also created */
```

## Simple Initialization

Before using a draw descriptor it needs to be initialized with the related function. For example, `lv_draw_label_dsc_init(&my_label_draw_dsc)`.

After initialization, each field of the draw descriptor can be set. The default values are quite sane and reasonable, so usually only a few fields need modification. Ad esempio:

```
/* In LV_EVENT_DRAW_MAIN */
lv_draw_label_dsc_t my_label_draw_dsc;
lv_draw_label_dsc_init(&my_label_draw_dsc);
my_label_draw_dsc.font = &my_font;
my_label_draw_dsc.color = lv_color_hex(0xffff0000);
my_label_draw_dsc.text = "Hello";

lv_area_t a = {10, 10, 200, 50}; /* Draw label here */

lv_draw_label(lv_event_get_layer(e), &my_label_draw_dsc, &a);
```

## Initialization for Widgets

When rendering a part of a Widget, helper functions can initialize draw descriptors based on the styles, and a specific Widget part in the current state.

Ad esempio:

```
/* In LV_EVENT_DRAW_MAIN */
lv_draw_rect_dsc_t cur_dsc;
lv_draw_rect_dsc_init(&cur_dsc);
lv_obj_init_draw_rect_dsc(obj, LV_PART_CURSOR, &cur_dsc);
cur_dsc.fill_color = lv_color_hex(0xffff0000); /* Modify if needed */
lv_draw_rect(layer, &cur_dsc, &area);
```

The `lv_obj_init_draw_...` functions automatically initialize the fields of the base descriptor.

## Modify the draw descriptors

In `LV_EVENT_DRAW_TASK_ADDED`, the draw descriptor of the `draw_task` can be accessed (using `lv_draw_task_get_label_dsc()` and similar functions) and modified (to change color, text, font, etc.). This means that in `LV_EVENT_DRAW_TASK_ADDED`, the `draw_tasks` and draw descriptors are already initialized and it's enough to change only a few specific values.

Ad esempio:

```
/* In LV_EVENT_DRAW_TASK_ADDED */
lv_draw_task_t * t = lv_event_get_draw_task(e);
lv_draw_label_dsc_t * draw_dsc = lv_draw_task_get_label_dsc(t);

/* Check a few things in `draw_dsc->base` */

/* Make the color lighter for longer texts */
draw_dsc->color = lv_color_lighten(draw_dsc->color,
                                     LV_MIN(lv_strlen(draw_dsc->text) * 5, 255));

/* Create new Draw Tasks if needed by calling
 * `lv_draw_...(draw_dsc->base.layer, ...)` functions */
```

## Rectangle Draw Descriptor

`lv_draw_rect_dsc_t` is a helper descriptor that combines:

- Fill
- Border [bordatura]
- Outline (a border with its own styles)
- Shadow
- Background image (an image with its own styles)

into a single call.

`lv_obj_init_draw_rect_dsc(obj, part, &dsc)`; initializes a draw descriptor from a Widget, and `lv_draw_rect(layer, &dsc, area)` draws the rectangle in a specified area.

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a rectangle to the canvas
 */
void lv_example_canvas_3(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);

    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_rect_dsc_t dsc;
    lv_draw_rect_dsc_init(&dsc);
    dsc.bg_color = lv_palette_main(LV_PALETTE_RED);
    dsc.border_color = lv_palette_main(LV_PALETTE_BLUE);
    dsc.border_width = 3;
    dsc.outline_color = lv_palette_main(LV_PALETTE_GREEN);
    dsc.outline_width = 2;
    dsc.outline_pad = 2;
    dsc.outline_opa = LV_OPA_50;
    dsc.radius = 5;
    dsc.border_width = 3;

    lv_area_t coords = {10, 10, 40, 30};

    lv_draw_rect(&layer, &dsc, &coords);
    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

## Fill Draw Descriptor

The main fields of `lv_draw_fill_dsc_t` are straightforward. It has a radius, opacity, and color to draw a rectangle. If opacity is 0, no draw task will be created.

- `lv_draw_fill_dsc_init(&dsc)` initializes a fill Draw Task.
- `lv_draw_sw_fill(layer, &dsc, area)` creates a Draw Task to fill an area.
- `lv_draw_task_get_fill_dsc(draw_task)` retrieves the fill descriptor from a Draw Task.

## Gradients

The `grad` field of the fill descriptor (or `lv_grad_dsc_t` in general) supports:

- Horizontal
- Vertical
- Skew
- Radial
- Conical

gradient types.

The following show some example gradients.

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the background style properties
 */
void lv_example_style_2(void)
{
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_radius(&style, 5);

    /*Make a gradient*/
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    static lv_grad_dsc_t grad;
    grad.dir = LV_GRAD_DIR_VER;
    grad.stops_count = 2;
    grad.stops[0].color = lv_palette_lighten(LV_PALETTE_GREY, 1);
    grad.stops[0].opa = LV_OPA_COVER;
    grad.stops[1].color = lv_palette_main(LV_PALETTE_BLUE);
```

(continues on next page)

(continua dalla pagina precedente)

```

grad.stops[1].opa = LV_OPA_COVER;
/*Shift the gradient to the bottom*/
grad.stops[0].frac  = 128;
grad.stops[1].frac  = 192;
lv_style_set_bg_grad(&style, &grad);
/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_center(obj);
}
#endif

```

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Simulate metallic knob using conical gradient
 * For best effect set LV_GRADIENT_MAX_STOPS to 8 or at least 3
 */
void lv_example_style_16(void)
{
#if LV_GRADIENT_MAX_STOPS >= 8
    static const lv_color_t grad_colors[8] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0xff, 0xff, 0xff),
        LV_COLOR_MAKE(0xfa, 0xfa, 0xfa),
        LV_COLOR_MAKE(0x79, 0x79, 0x79),
        LV_COLOR_MAKE(0x48, 0x48, 0x48),
        LV_COLOR_MAKE(0x4b, 0x4b, 0x4b),
        LV_COLOR_MAKE(0x70, 0x70, 0x70),
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
    };
#elif LV_GRADIENT_MAX_STOPS >= 3
    static const lv_color_t grad_colors[3] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0xff, 0xff, 0xff),
        LV_COLOR_MAKE(0x79, 0x79, 0x79),
    };
#else
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
        LV_COLOR_MAKE(0x79, 0x79, 0x79),
    };
#endif

/*Create a style with gradient background and shadow*/
static lv_style_t style;
lv_style_init(&style);
lv_style_set_radius(&style, 500);
lv_style_set_bg_opa(&style, LV_OPA_COVER);
lv_style_set_shadow_color(&style, lv_color_black());
lv_style_set_shadow_width(&style, 50);
lv_style_set_shadow_offset_x(&style, 20);
lv_style_set_shadow_offset_y(&style, 20);
lv_style_set_shadow_opa(&style, LV_OPA_50);

/*First define a color gradient. In this example we use a gray color map with random values.*/
static lv_grad_dsc_t grad;

lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

/*Make a conical gradient with the center in the middle of the object*/
#if LV_GRADIENT_MAX_STOPS >= 8
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 0, 120, LV_GRAD_EXTEND_REFLECT);
#elif LV_GRADIENT_MAX_STOPS >= 3
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 125, LV_GRAD_EXTEND_REFLECT);
#else
    lv_grad_conical_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, 45, 110, LV_GRAD_EXTEND_REFLECT);
#endif

/*Set gradient as background*/
lv_style_set_bg_grad(&style, &grad);

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_set_size(obj, 200, 200);
lv_obj_center(obj);
}

#else

void lv_example_style_16(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Using radial gradient as background
 */

```

(continues on next page)

(continua dalla pagina precedente)

```

void lv_example_style_17(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x9B, 0x18, 0x42),
        LV_COLOR_MAKE(0x00, 0x00, 0x00),
    };

    int32_t width = lv_display_get_horizontal_resolution(NULL);
    int32_t height = lv_display_get_vertical_resolution(NULL);

    static lv_style_t style;
    lv_style_init(&style);

    /*First define a color gradient. In this example we use a purple to black color map.*/
    static lv_grad_dsc_t grad;
    lv_grad_init_stops(&grad, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

    /*Make a radial gradient with the center in the middle of the object, extending to the farthest corner*/
    lv_grad_radial_init(&grad, LV_GRAD_CENTER, LV_GRAD_CENTER, LV_GRAD_RIGHT, LV_GRAD_BOTTOM, LV_GRAD_EXTEND_PAD);

    /*Set gradient as background*/
    lv_style_set_bg_grad(&style, &grad);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_set_size(obj, width, height);
    lv_obj_center(obj);
}

#else

void lv_example_style_17(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

/***
 * Using various gradients for button background
 */
void lv_example_style_18(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0x26, 0xa0, 0xda),
        LV_COLOR_MAKE(0x31, 0x47, 0x55),
    };

    /*Create a linear gradient going from the top left corner to the bottom at an angle, with reflected color map*/
    static lv_style_t style_with_linear_gradient_bg;
    static lv_grad_dsc_t linear_gradient_dsc; /*NOTE: the gradient descriptor must be static or global variable!*/

    lv_style_init(&style_with_linear_gradient_bg);
    lv_grad_init_stops(&linear_gradient_dsc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_linear_init(&linear_gradient_dsc, lv_pct(0), lv_pct(0), lv_pct(20), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_linear_gradient_bg, &linear_gradient_dsc);
    lv_style_set_bg_opa(&style_with_linear_gradient_bg, LV_OPA_COVER);

    /*Create a radial gradient with the center in the top left 1/3rd of the object, extending to the bottom right corner, with reflected color map*/
    static lv_style_t style_with_radial_gradient_bg;
    static lv_grad_dsc_t radial_gradient_dsc; /*NOTE: the gradient descriptor must be static or global variable!*/

    lv_style_init(&style_with_radial_gradient_bg);
    lv_grad_init_stops(&radial_gradient_dsc, grad_colors, NULL, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_radial_init(&radial_gradient_dsc, lv_pct(30), lv_pct(30), lv_pct(100), lv_pct(100), LV_GRAD_EXTEND_REFLECT);
    lv_style_set_bg_grad(&style_with_radial_gradient_bg, &radial_gradient_dsc);
    lv_style_set_bg_opa(&style_with_radial_gradient_bg, LV_OPA_COVER);

    /*Create buttons with different gradient styles*/

    lv_obj_t * btn;
    lv_obj_t * label;

    /*Simple horizontal gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
    lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
    lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_HOR, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -100);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Horizontal");
    lv_obj_center(label);

    /*Simple vertical gradient*/
    btn = lv_button_create(lv_screen_active());
    lv_obj_set_style_bg_color(btn, grad_colors[0], 0);
    lv_obj_set_style_bg_grad_color(btn, grad_colors[1], 0);
    lv_obj_set_style_bg_grad_dir(btn, LV_GRAD_DIR_VER, 0);
    lv_obj_set_size(btn, 150, 50);
    lv_obj_align(btn, LV_ALIGN_CENTER, 0, -40);

    label = lv_label_create(btn);
    lv_label_set_text(label, "Vertical");
    lv_obj_center(label);

    /*Complex linear gradient*/

```

(continues on next page)

(continua dalla pagina precedente)

```

btn = lv_button_create(lv_screen_active());
lv_obj_add_style(btn, &style_with_linear_gradient_bg, 0);
lv_obj_set_size(btn, 150, 50);
lv_obj_align(btn, LV_ALIGN_CENTER, 0, 20);

label = lv_label_create(btn);
lv_label_set_text(label, "Linear");
lv_obj_center(label);

/*Complex radial gradient*/
btn = lv_button_create(lv_screen_active());
lv_obj_add_style(btn, &style_with_radial_gradient_bg, 0);
lv_obj_set_size(btn, 150, 50);
lv_obj_align(btn, LV_ALIGN_CENTER, 0, 80);

label = lv_label_create(btn);
lv_label_set_text(label, "Radial");
lv_obj_center(label);
}

#else

void lv_example_style_18(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_width(label, LV_PCT(80));
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS is not enabled");
    lv_label_set_long_mode(label, LV_LABEL_LONG_MODE_SCROLL_CIRCULAR);
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif /*LV_BUILD_EXAMPLES*/

```

For each gradient type, multiple color and opacity values can be assigned. These are called "stops". The maximum number of stops is limited to [LV\\_GRADIENT\\_MAX\\_STOPS](#).

A gradient is basically a transition of colors and opacities between stops.

Besides just setting the color and opacity of each stop, it is also possible to set where they start relative to the whole gradient area.

For example with 3 stops it can be set like this:

- 10% red: 0--10% fully red
- 60% green: 10--60% transition from red to green, 60% is fully green
- 65% blue: fast transition from green to blue between 60%--65%. After 65% fully blue.

The position of the stops are called fractions or offsets and are 8 bit values where 0 is 0% and 255 is 100% of the whole gradient area.

`lv_grad_init_stops(grad_dsc, colors, opas, fracs, cnt)` initializes a gradient descriptor with stops containing the color, opacity and fraction of each stop.

```

static const lv_color_t colors[2] = {
    LV_COLOR_MAKE(0xe8, 0xe8, 0xe8),
    LV_COLOR_MAKE(0x79, 0x79, 0x79),
};

static const lv_opa_t opas[2] = {
    170,
    255,
};

static const uint8_t fracs[2] = {
    170,
    255,
};

lv_grad_init_stops(&grad, colors, opas, fracs, sizeof(colors) / sizeof(lv_color_t));

```

If the opacity array is `NULL` 255 will be used for each stop. If the fractions array is `NULL` the colors will be distributed evenly. For example with 3 colors: 0%, 50%, 100%

## Padding [riempimento]

Linear, radial, and conic gradients are defined between two points or angles. You can define how to pad the areas outside of the start and end points or angles:

- [LV\\_GRAD\\_EXTEND\\_PAD](#): Repeat the same color
- [LV\\_GRAD\\_EXTEND\\_REPEAT](#): Repeat the pattern
- [LV\\_GRAD\\_EXTEND\\_REFLECT](#): Repeat the pattern normally and mirrored alternately

## Horizontal and Vertical Gradients

The simplest and usually fastest gradient types are horizontal and vertical gradients.

After initializing the stops with `lv_grad_init_stops` call either `lv_grad_horizontal_init(&grad_dsc)` or `lv_grad_vertical_init(&grad_dsc)` to get a horizontal or vertical gradient descriptor.

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

static void position_bullet(lv_event_t * e, lv_point_t * p)
{
    lv_indev_t * indev = (lv_indev_t *) lv_event_get_param(e);
    lv_indev_get_point(indev, p);

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);

    p->x -= lv_obj_get_x(parent);
    p->y -= lv_obj_get_y(parent);

    int32_t w = lv_obj_get_width(parent);
    int32_t h = lv_obj_get_height(parent);
    lv_obj_set_pos(bullet, LV_CLAMP(5, p->x, w - 20), LV_CLAMP(5, p->y, h - 20));
}

static void frac_1_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    if(lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v) != LV_STYLE_RES_FOUND) {
        LV_LOG_WARN("style prop not found");
    }
    else {
        lv_grad_dsc_t * dsc = (lv_grad_dsc_t *)v.ptr;

        lv_point_t p;
        position_bullet(e, &p);

        lv_obj_t * bullet = lv_event_get_target_obj(e);
        lv_obj_t * parent = lv_obj_get_parent(bullet);
        dsc->stops[0].frac = (uint8_t)LV_CLAMP(0, p.x * 255 / lv_obj_get_width(parent), 255);

        lv_obj_invalidate(parent);
    }
}

static void frac_2_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    if(lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v) != LV_STYLE_RES_FOUND) {
        LV_LOG_WARN("style prop not found");
    }
    else {
        lv_grad_dsc_t * dsc = (lv_grad_dsc_t *)v.ptr;

        lv_point_t p;
        position_bullet(e, &p);

        lv_obj_t * bullet = lv_event_get_target_obj(e);
        lv_obj_t * parent = lv_obj_get_parent(bullet);

        dsc->stops[1].frac = (uint8_t)LV_CLAMP(0, p.x * 255 / lv_obj_get_width(parent), 255);
        lv_obj_invalidate(parent);
    }
}

/**
 * Play with a simple horizontal gradient.
 * Adjust the stop positions of the gradient.
 */
void lv_example_grad_1(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0xff, 0x00, 0x00),
        LV_COLOR_MAKE(0x00, 0xff, 0x00),
    };

    static const lv_opa_t grad_opa[2] = {
        LV_OPA_100,
        LV_OPA_0,
    };

    static const uint8_t frac[2] = {
        20 * 255 / 100, /*20%*/
        80 * 255 / 100, /*80%*/
    };

    static lv_style_t style;
    lv_style_init(&style);

    static lv_grad_dsc_t grad_dsc;
    lv_grad_init_stops(&grad_dsc, grad_colors, grad_opa, frac, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_horizontal_init(&grad_dsc);

    /*Set gradient as background*/
    lv_style_set_bg_grad(&style, &grad_dsc);
    lv_style_set_border_width(&style, 2);
    lv_style_set_pad_all(&style, 0);
    lv_style_set_radius(&style, 12);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_set_size(obj, lv_pct(80), lv_pct(80));
    lv_obj_center(obj);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * frac_1 = lv_button_create(obj);
lv_obj_set_size(frac_1, 15, 15);
lv_obj_set_style_bg_color(frac_1, lv_color_hex(0xffff00), 0);
lv_obj_add_event_cb(frac_1, frac_1_event_cb, LV_EVENT_PRESSING, &style);
lv_obj_set_ext_click_area(frac_1, 5);
lv_obj_set_pos(frac_1, lv_pct(20), lv_pct(50));

lv_obj_t * frac_2 = lv_button_create(obj);
lv_obj_set_size(frac_2, 15, 15);
lv_obj_set_style_bg_color(frac_2, lv_color_hex(0xffff00), 0);
lv_obj_add_event_cb(frac_2, frac_2_event_cb, LV_EVENT_PRESSING, &style);
lv_obj_set_ext_click_area(frac_2, 5);
lv_obj_set_pos(frac_2, lv_pct(80), lv_pct(50));
}

#endif

```

## Linear Gradients

The linear (or skew) gradient are similar to horizontal or vertical gradient but the angle of the gradient can be controlled.

The linear gradient will be rendered along a line defined by 2 points.

After initializing the stops with `lv_grad_init_stops()` call `lv_grad_linear_init(&grad_dsc, from_x, from_y, to_x, to_y, LV_GRAD_EXTEND_...)` with your point values and extend pattern strategy to get a linear gradient descriptor.

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL
#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS
static void position_bullet(lv_event_t * e, lv_point_t * p)
{
    lv_indev_t * indev = (lv_indev_t *) lv_event_get_param(e);
    lv_indev_get_point(indev, p);

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);

    p->x -= lv_obj_get_x(parent);
    p->y -= lv_obj_get_y(parent);

    int32_t w = lv_obj_get_width(parent);
    int32_t h = lv_obj_get_height(parent);
    lv_obj_set_pos(bullet, LV_CLAMP(5, p->x, w - 20), LV_CLAMP(5, p->y, h - 20));
}

static void start_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *)v.ptr;

    lv_point_t p;
    position_bullet(e, &p);

    dsc->params.linear.start.x = p.x;
    dsc->params.linear.start.y = p.y;

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);
    lv_obj_invalidate(parent);
}

static void end_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *)v.ptr;

    lv_point_t p;
    position_bullet(e, &p);

    dsc->params.linear.end.x = p.x;
    dsc->params.linear.end.y = p.y;

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);
    lv_obj_invalidate(parent);
}

/**
 * Play with the linear gradient.
 * Adjust the 2 point in between the a linear gradient can be drawn (can be skew as well)
 */
void lv_example_grad_2(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0xff, 0x00, 0x00),
        LV_COLOR_MAKE(0x00, 0xff, 0x00),
    };

    static const lv_opa_t grad_opa[2] = {
        LV_OPA_100,
        LV_OPA_0,
    };

    static lv_style_t style;
    lv_style_init(&style);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

/*First define a color gradient. In this example we use a purple to black color map.*/
static lv_grad_dsc_t grad;

lv_grad_init_stops(&grad, grad_colors, grad_opa, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

lv_grad_linear_init(&grad, 100, 100, 200, 200, 150, LV_GRAD_EXTEND_PAD);

/*Set gradient as background*/
lv_style_set_bg_grad(&style, &grad);
lv_style_set_border_width(&style, 2);
lv_style_set_pad_all(&style, 0);
lv_style_set_radius(&style, 12);

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_set_size(obj, lv_pct(80), lv_pct(80));
lv_obj_center(obj);

lv_obj_t * start = lv_button_create(obj);
lv_obj_set_size(start, 15, 15);
lv_obj_set_style_bg_color(start, lv_color_hex(0x0000ff), 0);
lv_obj_add_event_cb(start, start_event_cb, LV_EVENT_PRESSING, &style);
lv_obj_set_ext_click_area(start, 5);
lv_obj_set_pos(start, 100, 100);

lv_obj_t * end = lv_button_create(obj);
lv_obj_set_size(end, 15, 15);
lv_obj_set_style_bg_color(end, lv_color_hex(0x00ffff), 0);
lv_obj_add_event_cb(end, end_event_cb, LV_EVENT_PRESSING, &style);
lv_obj_set_ext_click_area(end, 5);
lv_obj_set_pos(end, 200, 150);
}

#else

void lv_example_grad_2(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS needs to be enabled");
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif

```

## Radial Gradients

The radial gradient is described by two circles: an outer circle and an inner circle (also called the focal point). The gradient will be calculated between the focal point's circle and the edge of the outer circle.

If the center of the focal point and the center of the main circle are the same, the gradient will spread evenly in all directions. If the center points are not the same, the gradient will have an egg shape.

The focal point's circle should be inside the main circle.

After initializing the stops with `lv_grad_init_stops()`, the outer circle can be set by: `lv_grad_radial_init(&grad_dsc, center_x, center_y, edge_x, edge_y, LV_GRAD_EXTEND_...)`

For both the center and edge coordinates, px or `lv_pct()` values can be used.

The inner circle (focal point) can be set with: `lv_grad_radial_set_focal(&grad_dsc, center_x, center_y, radius)`

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS

static void position_bullet(lv_event_t * e, lv_point_t * p)
{
    lv_indev_t * indev = (lv_indev_t *) lv_event_get_param(e);
    lv_indev_get_point(indev, p);

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);

    p->x -= lv_obj_get_x(parent);
    p->y -= lv_obj_get_y(parent);

    int32_t w = lv_obj_get_width(parent);
    int32_t h = lv_obj_get_height(parent);
    lv_obj_set_pos(bullet, LV_CLAMP(5, p->x, w - 20), LV_CLAMP(5, p->y, h - 20));
}

static void focal_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *) v.ptr;

    lv_point_t p;
    position_bullet(e, &p);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

dsc->params.radial.focal.x = p.x;
dsc->params.radial.focal.y = p.y;
dsc->params.radial.focal_extent.x = p.x + 10;
dsc->params.radial.focal_extent.y = p.y;

lv_obj_t * bullet = lv_event_get_target_obj(e);
lv_obj_t * parent = lv_obj_get_parent(bullet);
lv_obj_invalidate(parent);
}

static void end_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *)v.ptr;

    lv_point_t p;
    position_bullet(e, &p);

    dsc->params.radial.end.x = p.x;
    dsc->params.radial.end.y = p.y;
    dsc->params.radial.end_extent.x = p.x + 100;
    dsc->params.radial.end_extent.y = p.y;
    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);
    lv_obj_invalidate(parent);
}

/**
 * Play with the radial gradient
 * Adjust the end circle and focal point position.
 * The radius of the end circle and an focal point are hardcoded in the example.
 */
void lv_example_grad_3(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0xff, 0x00, 0x00),
        LV_COLOR_MAKE(0x00, 0xff, 0x00),
    };

    static const lv_opa_t grad_opa[2] = {
        LV_OPA_100,
        LV_OPA_0,
    };

    static lv_style_t style;
    lv_style_init(&style);

    /*First define a color gradient. In this example we use a purple to black color map.*/
    static lv_grad_dsc_t grad;
    lv_grad_init_stops(&grad, grad_colors, grad_opa, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

    /*Init a radial gradient where the center is at 100;100
     *and the edge of the circle is at 200;100.
     *Try LV_GRAD_EXTEND_REFLECT and LV_GRAD_EXTEND_REPEAT too. */
    lv_grad_radial_init(&grad, 100, 100, 200, 100, LV_GRAD_EXTEND_PAD);

    /*The gradient will be calculated between the focal point's circle and the
     *edge of the circle. If the center of the focal point and the
     *center of the main circle is the same, the gradient will spread
     *evenly in all directions. The focal point should be inside the
     *main circle.*/
    lv_grad_radial_set_focal(&grad, 50, 50, 10);

    /*Set the widget containing the gradient*/
    lv_style_set_bg_grad(&style, &grad);
    lv_style_set_border_width(&style, 2);
    lv_style_set_pad_all(&style, 0);
    lv_style_set_radius(&style, 12);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_set_size(obj, lv_pct(80), lv_pct(80));
    lv_obj_center(obj);

    lv_obj_t * focal = lv_button_create(obj);
    lv_obj_set_size(focal, 15, 15);
    lv_obj_set_style_bg_color(focal, lv_color_hex(0x0000ff), 0);
    lv_obj_add_event_cb(focal, focal_event_cb, LV_EVENT_PRESSING, &style);
    lv_obj_set_ext_click_area(focal, 5);
    lv_obj_set_pos(focal, 50, 50);

    lv_obj_t * end = lv_button_create(obj);
    lv_obj_set_size(end, 15, 15);
    lv_obj_set_style_bg_color(end, lv_color_hex(0x00ffff), 0);
    lv_obj_add_event_cb(end, end_event_cb, LV_EVENT_PRESSING, &style);
    lv_obj_set_ext_click_area(end, 5);
    lv_obj_set_pos(end, 100, 100);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif

```

## Conic Gradients

The conic gradient is defined between the angles of a circle, and colors are mapped to each angle.

After initializing the stops with `lv_grad_init_stops()`, the conic gradient can be set up with: `lv_grad_conical_init(&grad, center_x, center_y, angle_start, angle_end, LV_GRAD_EXTEND_...)`

For both the center and edge coordinates, `px` or `lv_pct()` values can be used.

The zero angle is on the right-hand side, and 90 degrees is at the bottom.

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL
#endif
#if LV_USE_DRAW_SW_COMPLEX_GRADIENTS
static void position_bullet(lv_event_t * e, lv_point_t * p)
{
    lv_indev_t * indev = (lv_indev_t *) lv_event_get_param(e);
    lv_indev_get_point(indev, p);

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);

    p->x -= lv_obj_get_x(parent);
    p->y -= lv_obj_get_y(parent);

    int32_t w = lv_obj_get_width(parent);
    int32_t h = lv_obj_get_height(parent);
    lv_obj_set_pos(bullet, LV_CLAMP(5, p->x, w - 20), LV_CLAMP(5, p->y, h - 20));
}

static void start_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *) v.ptr;

    lv_point_t p;
    position_bullet(e, &p);

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);
    p.x -= lv_obj_get_width(parent) / 2;
    p.y -= lv_obj_get_height(parent) / 2;

    dsc->params.conical.start_angle = lv_atan2(p.y, p.x);
    lv_obj_invalidate(parent);
}

static void end_event_cb(lv_event_t * e)
{
    lv_style_t * style = (lv_style_t *) lv_event_get_user_data(e);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_BG_GRAD, &v);
    lv_grad_dsc_t * dsc = (lv_grad_dsc_t *) v.ptr;

    lv_point_t p;
    position_bullet(e, &p);

    lv_obj_t * bullet = lv_event_get_target_obj(e);
    lv_obj_t * parent = lv_obj_get_parent(bullet);
    p.x -= lv_obj_get_width(parent) / 2;
    p.y -= lv_obj_get_height(parent) / 2;

    dsc->params.conical.end_angle = lv_atan2(p.y, p.x);
    lv_obj_invalidate(parent);
}

/**
 * Play with the conical gradient
 */
void lv_example_grad_4(void)
{
    static const lv_color_t grad_colors[2] = {
        LV_COLOR_MAKE(0xff, 0x00, 0x00),
        LV_COLOR_MAKE(0x00, 0xff, 0x00),
    };

    static const lv_opa_t grad_opa[2] = {
        LV_OPA_100,
        LV_OPA_0,
    };

    static lv_style_t style;
    lv_style_init(&style);

    /*First define a color gradient. In this example we use a purple to black color map.*/
    static lv_grad_dsc_t grad;
    lv_grad_init_stops(&grad, grad_colors, grad_opa, NULL, sizeof(grad_colors) / sizeof(lv_color_t));

    lv_grad_init_stops(&grad, grad_colors, grad_opa, NULL, sizeof(grad_colors) / sizeof(lv_color_t));
    lv_grad_conical_init(&grad, lv_pct(50), lv_pct(50), 0, 180, LV_GRAD_EXTEND_PAD);

    /*Set gradient as background*/
    lv_style_set_bg_grad(&style, &grad);
    lv_style_set_border_width(&style, 2);
    lv_style_set_pad_all(&style, 0);
    lv_style_set_radius(&style, 12);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_set_size(obj, lv_pct(80), lv_pct(80));
    lv_obj_center(obj);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * start = lv_button_create(obj);
lv_obj_set_size(start, 15, 15);
lv_obj_set_style_bg_color(start, lv_color_hex(0x0000ff), 0);
lv_obj_add_event_cb(start, start_event_cb, LV_EVENT_PRESSING, &style);
lv_obj_set_ext_click_area(start, 5);
lv_obj_set_pos(start, lv_pct(80), lv_pct(50));

lv_obj_t * end = lv_button_create(obj);
lv_obj_set_size(end, 15, 15);
lv_obj_set_style_bg_color(end, lv_color_hex(0x00ffff), 0);
lv_obj_add_event_cb(end, end_event_cb, LV_EVENT_PRESSING, &style);
lv_obj_set_ext_click_area(end, 5);
lv_obj_set_pos(end, lv_pct(20), lv_pct(50));
}

#else

void lv_example_grad_4(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "LV_USE_DRAW_SW_COMPLEX_GRADIENTS needs to be enabled");
    lv_obj_center(label);
}

#endif /*LV_USE_DRAW_SW_COMPLEX_GRADIENTS*/
#endif

```

## Border Draw Descriptor

The `lv_draw_border_dsc_t` border descriptor has radius, opacity, width, color, and side fields. If the opacity or width is 0, no Draw Task will be created.

`side` can contain ORed values of `lv_border_side_t`, such as `LV_BORDER_SIDE_BOTTOM`. `LV_BORDER_SIDE_ALL` applies to all sides, while `LV_BORDER_SIDE_INTERNAL` is used by higher layers (e.g. a table Widget) to calculate border sides. However, the drawing routine receives only simpler values.

The following functions are used for border drawing:

- `lv_draw_border_dsc_init(&dsc)` initializes a border Draw Task.
- `lv_draw_sw_border(layer, &dsc, area)` creates a Draw Task to draw a border inward from its area.
- `lv_draw_task_get_border_dsc(draw_task)` retrieves the border descriptor from a Draw Task.

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the border style properties
 */
void lv_example_style_3(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 10);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add border to the bottom+right*/
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_border_width(&style, 5);
    lv_style_set_border_opa(&style, LV_OPA_50);
    lv_style_set_border_side(&style, (lv_border_side_t)(LV_BORDER_SIDE_BOTTOM | LV_BORDER_SIDE_RIGHT));

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif

```

## Outlines

The outline is similar to the border but is drawn outside the object's draw area.

In practice, there is no dedicated outline descriptor like `lv_draw_outline_dsc_t`, because from the rendering perspective, the outline is simply another border rendered outside the object's bounds.

The outline is used only in `lv_draw_rect_dsc_t` for convenience. The two differences compared to borders in `lv_draw_rect_dsc_t` are:

- There is an `outline_pad` property to specify the gap between the target area and the inner side of the outline. It can be negative. For example, if `outline_pad = -width`, the outline will resemble a border.

- There is no `border_side` property for the outline. It's always rendered as a full rectangle.

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the outline style properties
 */
void lv_example_style_4(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

    /*Add outline*/
    lv_style_set_outline_width(&style, 2);
    lv_style_set_outline_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_outline_pad(&style, 8);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_obj_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}

#endif
```

## Box Shadow Draw Descriptor

The `lv_draw_box_shadow_dsc_t` box shadow descriptor describes a **rounded rectangle-shaped shadow**. It cannot generate shadows for arbitrary shapes, text, or images. It includes the following fields:

### **radius**

Radius, `LV_RADIUS_CIRCLE`.

### **color**

Shadow color.

### **width**

Shadow width (blur radius).

### **spread**

Expands the rectangle in all directions; can be negative.

### **ofs\_x**

Horizontal offset.

### **ofs\_y**

Vertical offset.

### **opa**

Opacity (0-255 range). Values like `LV_OPA_TRANSP`, `LV_OPA_10`, etc., can also be used.

### **bg\_cover**

Set to 1 if the background will cover the shadow (a hint for the renderer to skip masking).

Note: Rendering large shadows may be slow or memory-intensive.

The following functions are used for box shadow drawing:

- `lv_draw_box_shadow_dsc_init(&dsc)` initializes a box shadow Draw Task.
- `lv_draw_sw_box_shadow(layer, &dsc, area)` creates a Draw Task for a rectangle's shadow. The shadow's size and position depend on the width, spread, and offset.
- `lv_draw_task_get_box_shadow_dsc(draw_task)` retrieves the box shadow descriptor from a Draw Task.

```
#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES

/***
 * Using the Shadow style properties
 */
void lv_example_style_5(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_style_set_radius(&style, 5);
lv_style_set_bg_opa(&style, LV_OPA_COVER);
lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 1));

/*Add a shadow*/
lv_style_set_shadow_width(&style, 55);
lv_style_set_shadow_color(&style, lv_palette_main(LV_PALETTE_BLUE));

/*Create an object with the new style*/
lv_obj_t * obj = lv_obj_create(lv_screen_active());
lv_obj_add_style(obj, &style, 0);
lv_obj_center(obj);
}

#endif

```

## Image Draw Descriptor

The `lv_draw_image_dsc_t` image descriptor defines the parameters for image drawing. It is a complex descriptor with the following options:

### **src**

The image source, either a pointer to `lv_image_dsc_t` or a file path.

### **opa**

Opacity in the 0--255 range. Options like `LV_OPA_TRANSP`, `LV_OPA_10`, etc., can also be used.

### **clip\_radius**

Clips the corners of the image with this radius. Use `LV_RADIUS_CIRCLE` for the maximum radius.

### **rotation**

Image rotation in 0.1-degree units (e.g., 234 means 23.4°).

### **scale\_x**

Horizontal scaling (zoom) of the image. 256 (`LV_SCALE_NONE`) means no zoom, 512 doubles the size, and 128 halves it.

### **scale\_y**

Same as `scale_x` but for vertical scaling.

### **skew\_x**

Horizontal skew (parallelogram-like transformation) in 0.1-degree units (e.g., 456 means 45.6°).

### **skew\_y**

Vertical skew, similar to `skew_x`.

### **pivot**

The pivot point for transformations (scaling and rotation). (0,0) is the top-left corner of the image and can be set outside the image.

### **bitmap\_mask\_src**

Pointer to an A8 or L8 image descriptor used to mask the image. The mask is always center-aligned.

### **recolor**

Mixes this color with the image. For `LV_COLOR_FORMAT_A8`, this will be the visible pixels' color.

### **recolor\_opa**

Intensity of recoloring (0 means no recoloring, 255 means full cover).

### **blend\_mode**

Defines how to blend image pixels with the background. See `lv_blend_mode_t` for more details.

### **antialias**

Set to 1 to enable anti-aliasing for transformations.

**tile**

Tiles the image (repeats it both horizontally and vertically) if the image is smaller than the **image\_area** field in **lv\_draw\_image\_dsc\_t**.

**image\_area**

Indicates the original, non-clipped area where the image is drawn. This is essential for:

1. Layer rendering, where only part of a layer may be rendered and **clip\_radius** needs the original image dimensions.
2. Tiling, where the draw area is larger than the image.

**sup**

Internal field to store information about the palette or color of A8 images.

Functions for image drawing:

- **lv\_draw\_image\_dsc\_init(&dsc)** initializes an image draw descriptor.
- **lv\_draw\_image(layer, &dsc, area)** creates a task to draw an image in a given area.
- **lv\_draw\_task\_get\_image\_dsc(draw\_task)** retrieves the image descriptor from a task.

```
#include "../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw an image to the canvas
 */
void lv_example_canvas_6(void)
{
    /*Create a buffer for the canvas*/
    static uint8_t cbuf[LV_CANVAS_BUF_SIZE(CANVAS_WIDTH, CANVAS_HEIGHT, 32, LV_DRAW_BUF_STRIDE_ALIGN)];

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    LV_IMAGE_DECLARE(img_star);
    lv_draw_image_dsc_t dsc;
    lv_draw_image_dsc_init(&dsc);
    dsc.src = &img_star;

    lv_area_t coords = {10, 10, 10 + img_star.header.w - 1, 10 + img_star.header.h - 1};

    lv_draw_image(&layer, &dsc, &coords);

    lv_canvas_finish_layer(canvas, &layer);
}
#endif
```

```
#include "../lv_examples.h"
#ifndef LV_BUILD_EXAMPLES || !LV_USE_IMAGE
#endif

/***
 * Using the Image style properties
 */
void lv_example_style_6(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    /*Set a background color and a radius*/
    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 3));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));

    lv_style_set_image_recolor(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_image_recolor_opa(&style, LV_OPA_50);
    lv_style_set_transform_rotation(&style, 300);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_image_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    LV_IMAGE_DECLARE(img_cogwheel_argb);
    lv_image_set_src(obj, &img_cogwheel_argb);

    lv_obj_center(obj);
}
#endif
```

## Layers - Special Images

Layers are treated as images, so an `lv_draw_image_dsc_t` can describe how layers are blended into their parent layers. All image features apply to layers as well.

`lv_draw_layer(layer, &dsc, area)` initializes the blending of a layer back to its parent layer. Additionally, image-drawing-related functions can be used for layers.

For more details, see [I Layer](#).

## Label Draw Descriptor

The `lv_draw_label_dsc_t` label descriptor provides extensive options for controlling text rendering:

### **text**

The text to render.

### **font**

Font to use, with support for fallback fonts.

### **color**

Text color.

### **opa**

Text opacity.

### **line\_space**

Additional space between lines.

### **letter\_space**

Additional space between characters.

### **ofs\_x**

Horizontal text offset.

### **ofs\_y**

Vertical text offset.

### **sel\_start**

Index of the first character for selection (not byte index). `LV_DRAW_LABEL_NO_TXT_SEL` means no selection.

### **sel\_end**

Index of the last character for selection.

### **sel\_color**

Color of selected characters.

### **sel\_bg\_color**

Background color for selected characters.

### **align**

Text alignment. See `lv_text_align_t`.

### **bidi\_dir**

Base direction for right-to-left text rendering (e.g., Arabic). See `lv_base_dir_t`.

### **decor**

Text decoration, e.g., underline. See `lv_text_decor_t`.

### **flag**

Flags for text rendering. See `lv_text_flag_t`.

### **text\_length**

Number of characters to render (0 means render until 0).

### **text\_local**

Set to 1 to allocate a buffer and copy the text.

**text\_static**

Indicates `text` is constant and its pointer can be cached.

**hint**

Pointer to externally stored data to speed up rendering. See `lv_draw_label_hint_t`.

Functions for text drawing:

- `lv_draw_label_dsc_init(&dsc)` initializes a label draw descriptor.
- `lv_draw_label(layer, &dsc, area)` creates a task to render text in an area.
- `lv_draw_character(layer, &dsc, point, unicode_letter)` creates a task to draw a character at a specific point.
- `lv_draw_task_get_label_dsc(draw_task)` retrieves the label descriptor from a task.

For character-specific drawing in draw units, use `lv_draw_label_iterate_characters(draw_unit, draw_dsc, area, callback)`. This iterates through all characters, calculates their positions, and calls the callback for rendering each character. For callback details, see `lv_draw_glyph_cb_t`.

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_FONT_MONTserrat_18 && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a text to the canvas
 */
void lv_example_canvas_4(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_label_dsc_t dsc;
    lv_draw_label_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.font = &lv_font_montserrat_18;
    dsc.decor = LV_TEXT_DECOR_UNDERLINE;
    dsc.text = "Hello";

    lv_area_t coords = {10, 10, 30, 60};
    lv_draw_label(&layer, &dsc, &coords);
    lv_canvas_finish_layer(canvas, &layer);
}

#endif
```

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LABEL

/***
 * Using the text style properties
 */
void lv_example_style_8(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_radius(&style, 5);
    lv_style_set_bg_opa(&style, LV_OPA_COVER);
    lv_style_set_bg_color(&style, lv_palette_lighten(LV_PALETTE_GREY, 2));
    lv_style_set_border_width(&style, 2);
    lv_style_set_border_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_pad_all(&style, 10);

    lv_style_set_text_color(&style, lv_palette_main(LV_PALETTE_BLUE));
    lv_style_set_text_letter_space(&style, 5);
    lv_style_set_text_line_space(&style, 20);
    lv_style_set_text_decor(&style, LV_TEXT_DECOR_UNDERLINE);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_label_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_label_set_text(obj, "Text of\n"
                        "a label");

    lv_obj_center(obj);
}

#endif
```

## Arc Draw Descriptor

The `lv_draw_arc_dsc_t` arc descriptor defines arc rendering with these fields:

### **color**

Arc color.

### **img\_src**

Image source for the arc, or `NULL` if unused.

### **width**

Arc thickness.

### **start\_angle**

Starting angle in degrees (e.g., 0° is 3 o'clock, 90° is 6 o'clock).

### **end\_angle**

Ending angle.

### **center**

Arc center point.

### **radius**

Arc radius.

### **opa**

Arc opacity (0--255).

### **rounded**

Rounds the arc ends.

Functions for arc drawing:

- `lv_draw_arc_dsc_init(&dsc)` initializes an arc descriptor.
- `lv_draw_arc(layer, &dsc)` creates a task to render an arc.
- `lv_draw_task_get_arc_dsc(draw_task)` retrieves arc descriptor from task.

```
#include "../../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw an arc to the canvas
 */
void lv_example_canvas_5(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xcc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_arc_dsc_t dsc;
    lv_draw_arc_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
    dsc.width = 5;
    dsc.center.x = 25;
    dsc.center.y = 25;
    dsc.radius = 15;
    dsc.start_angle = 0;
    dsc.end_angle = 220;

    lv_draw_arc(&layer, &dsc);

    lv_canvas_finish_layer(canvas, &layer);
}

#endif
```

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_ARC

/***
 * Using the Arc style properties
 */
void lv_example_style_7(void)
```

(continues on next page)

(continua dalla pagina precedente)

```
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_arc_color(&style, lv_palette_main(LV_PALETTE_RED));
    lv_style_set_arc_width(&style, 4);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_arc_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);
    lv_obj_center(obj);
}
#endif
```

## Line Draw Descriptor

The `lv_draw_line_dsc_t` line descriptor defines line rendering with these fields:

### **p1**

First point of line (supports floating-point coordinates).

### **p2**

Second point of line (supports floating-point coordinates).

### **color**

Line color.

### **width**

Line thickness.

### **opa**

Line opacity (0--255).

### **dash\_width**

Length of dashes (0 means no dashes).

### **dash\_gap**

Length of gaps between dashes (0 means no dashes).

### **round\_start**

Rounds the line start.

### **round\_end**

Rounds the line end.

### **raw\_end**

Set to 1 to skip end calculations if they are unnecessary.

Functions for line drawing:

- `lv_draw_line_dsc_init(&dsc)` initializes a line descriptor.
- `lv_draw_line(layer, &dsc)` creates a task to draw a line.
- `lv_draw_task_get_line_dsc(draw_task)` retrieves line descriptor.

```
#include "../../../lv_examples.h"
#if LV_USE_CANVAS&& LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/***
 * Draw a line to the canvas
 */
void lv_example_canvas_7(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_line_dsc_t dsc;
    lv_draw_line_dsc_init(&dsc);
    dsc.color = lv_palette_main(LV_PALETTE_RED);
```

(continues on next page)

(continua dalla pagina precedente)

```

dsc.width = 4;
dsc.round_end = 1;
dsc.round_start = 1;
dsc.p1.x = 15;
dsc.p1.y = 15;
dsc.p2.x = 35;
dsc.p2.y = 10;
lv_draw_line(&layer, &dsc);

lv_canvas_finish_layer(canvas, &layer);
}

#endif

```

```

#include "../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_LINE

/***
 * Using the line style properties
 */
void lv_example_style_9(void)
{
    static lv_style_t style;
    lv_style_init(&style);

    lv_style_set_line_color(&style, lv_palette_main(LV_PALETTE_GREY));
    lv_style_set_line_width(&style, 6);
    lv_style_set_line_rounded(&style, true);

    /*Create an object with the new style*/
    lv_obj_t * obj = lv_line_create(lv_screen_active());
    lv_obj_add_style(obj, &style, 0);

    static lv_point_precise_t p[] = {{10, 30}, {30, 50}, {100, 0}};
    lv_line_set_points(obj, p, 3);

    lv_obj_center(obj);
}
#endif

```

## Triangle Draw Descriptor

Triangles are defined by `lv_draw_triangle_dsc_t`, which includes:

**p[3]**

3 points for the triangle's vertices.

**color**

Triangle color.

**opa**

Triangle opacity.

**grad**

Gradient options. If `grad.dir` is not `LV_GRAD_DIR_NONE`, the `color` field is ignored. The `opa` field adjusts overall opacity.

Functions for triangle drawing: - `lv_draw_triangle_dsc_init(&dsc)` initializes a triangle descriptor. - `lv_draw_triangle(layer, &dsc)` creates a task to draw a triangle. - `lv_draw_task_get_triangle_dsc(draw_task)` retrieves triangle descriptor.

```

#include "../lv_examples.h"
#if LV_USE_CANVAS && LV_BUILD_EXAMPLES

#define CANVAS_WIDTH 150
#define CANVAS_HEIGHT 150

/***
 * Draw a triangle to the canvas
 */
void lv_example_canvas_9(void)
{
    /*Create a buffer for the canvas*/
    LV_DRAW_BUF_DEFINE_STATIC(draw_buf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_COLOR_FORMAT_ARGB8888);
    LV_DRAW_BUF_INIT_STATIC(draw_buf);

    /*Create a canvas and initialize its palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_screen_active());
    lv_canvas_set_draw_buf(canvas, &draw_buf);
    lv_canvas_fill_bg(canvas, lv_color_hex3(0xccc), LV_OPA_COVER);
    lv_obj_center(canvas);

    lv_layer_t layer;
    lv_canvas_init_layer(canvas, &layer);

    lv_draw_triangle_dsc_t tri_dsc;
    lv_draw_triangle_dsc_init(&tri_dsc);
    tri_dsc.p[0].x = 10;
    tri_dsc.p[0].y = 10;
    tri_dsc.p[1].x = 100;
    tri_dsc.p[1].y = 30;
    tri_dsc.p[2].x = 50;

```

(continues on next page)

(continua dalla pagina precedente)

```

tri_dsc.p[2].y = 100;
tri_dsc.grad.stops_count = 2;
tri_dsc.grad.dir = LV_GRAD_DIR_VER;
tri_dsc.grad.stops[0].color = lv_color_hex(0xffff0000);
tri_dsc.grad.stops[0].frac = 64; /*Start at 25%*/
tri_dsc.grad.stops[0].opa = LV_OPA_COVER;
tri_dsc.grad.stops[1].color = lv_color_hex(0x0000ff);
tri_dsc.grad.stops[1].opa = LV_OPA_TRANSP;
tri_dsc.grad.stops[1].frac = 3 * 64; /*End at 75%*/

tri_dsc.opa = 128; /*Set the overall opacity to 50%*/

lv_draw_triangle(&layer, &tri_dsc);

lv_canvas_finish_layer(canvas, &layer);

}

#endif

```

## Vector Draw Descriptor

TODO [da fare]

## Masking Operation

There are several options to mask parts of a layer, Widget, or drawing:

- Radius of Rectangles:** Set the `radius` style property or the `radius` in the draw descriptors. This creates rounded rectangles, borders, outlines, etc.. However, the content of subsequent renderings will not be masked out in the corners.
- Clip Radius of Images:** Similar to rectangles, images can also be rendered with a `radius`. Since layer drawing and image drawing are handled the same way, this works for layers as well.

You can draw various content on a layer and then render the layer with a `clip_radius`, masking out all the content on the corners.

- Rectangle Mask Draw Task:** A special Draw Task can mask out a rectangle from a layer by setting the alpha channel of certain pixels to 0. To achieve this:

- Create an `lv_draw_mask_rect_dsc_t` descriptor.
- Set `area`, `radius`, and `keep_outside` parameters. If `keep_outside` is set to 1, areas outside of `area` remain unchanged. Otherwise, they are cleared.
- Call `lv_draw_mask_rect(layer, &dsc)`.

Note: The layer must have a color format with an alpha channel, typically `LV_COLOR_FORMAT_ARGB8888`.

In most cases, the "*Clip Radius of Images*" method is better because it blends the layer with a radius mask on the fly, avoiding a dedicated masking step. However, the "*Rectangle Mask Draw Task*" is useful when multiple areas need clearing or when the area to be masked differs from the layer area.

- Clip Corner Style Property:** Enabling `..._style_clip_corner` in a local or global style allows LVGL to create a layer for the top and bottom corner areas of a Widget. It renders the children there and blends it by setting `clip_radius` to the layer.
- Bitmap Masking for Images:** Using `..._style_bitmap_mask` or `bitmap_mask` in `lv_draw_image_dsc_t` allows setting an A8 or L8 image as a mask for an image/layer during blending.
  - Limitation: The mask always aligns to the center, and only one bitmap mask can be used for an image/layer.
  - When `..._style_bitmap_mask` is used, LVGL automatically creates a layer, renders the Widgets there, and applies the bitmap mask during blending.
  - Alternatively, the `bitmap_mask` property in the draw descriptor can be used directly for image drawing.

By using the Canvas Widget with an `LV_COLOR_FORMAT_L8` buffer, bitmap masks can be rendered dynamically.

## API

*lv\_draw.h*  
*lv\_draw\_arc.h*  
*lv\_draw\_image.h*  
*lv\_draw\_label.h*  
*lv\_draw\_line.h*  
*lv\_draw\_mask.h*  
*lv\_draw\_rect.h*  
*lv\_draw\_triangle.h*  
*lv\_grad.h*

## XML - Declarative UI

### 8.1 Panoramica

#### 8.1.1 Introduzione

The LVGL XML Module implements LVGL's Declarative UI by loading UI elements written in XML. The XML file can be written by hand, but it's highly recommended to use [LVGL's UI editor](#) to write the XML files. This UI editor provides features like:

- Instant preview of the XML files
- Autocomplete and Syntax highlighting
- Online preview for collaboration and testing
- [Figma](#) integration to easily reimplement Figma designs

#### Avvertimento

The UI editor and the XML loader are still under development and not production-ready. Consider them as an open beta or experimental features.

Describing the UI in XML in a declarative manner offers several advantages:

- XML files can be loaded at runtime (e.g. from an SD card) allowing you to change the application's UI without changing the firmware.
- XML files can be loaded dynamically as well, which allows for things like creating UIs from plugins or apps that are downloaded by the firmware.
- XML is simpler to write than C, enabling people with different skill sets to create LVGL UIs.
- XML is textual data, making it easy to parse and manipulate with a large number of programming and scripting languages.
- XML can be used to generate LVGL code in any language.
- XML helps to separate the view from the internal logic.
- The XML syntax uses the same properties as the C API, so it's easy to learn. E.g. style properties like `bg_color`, `line_width`, or widget properties like `width`, `height`, `label text`, etc.

- A powerful data binding engine allows you to easily bind the widgets to global data.

## Concept

The XML files are Component-oriented. To be more specific, they are **Component - Library** oriented. That is, they are structured in a way to make it easy to create reusable Component Libraries.

For example, a company can have a Component Library for the basic Widgets for all its products (generic widgets for all smart devices), and create other industry-specific Libraries (smart-home specific, smart-watch specific, etc.) containing only a few extra Widgets and Components.

These Component Libraries are independent, can be reused across many products, and can be independently versioned and managed. Imagine a Component Library as a collection of XML files that describe Widgets, Components, Screens, images, fonts, and other assets stored in a git repository, which can be a submodule in many projects. If someone finds a bug in the Component Library, they can simply fix it in one place and push it back to the git repository so that other projects can be updated from it.

The built-in Widgets of LVGL are considered **the core Component Library** which is always available.

A UI editor project can have any number of Component Libraries but will always have at least 2:

- LVGL's built-in Widgets, and
- XML-based definitions of Screen contents, along with other project-specific Components.

## Widgets, Components, and Screens

It is important to distinguish between *Widgets*, *Components*, and *Screens*.

### I Widget

*Widgets* are the core building blocks of the UI and are **not meant to be loaded at runtime** but rather compiled into the application as C code. The main characteristics of Widgets are:

- In XML, they start with a `<widget>` root element.
- They are similar to LVGL's built-in Widgets.
- They are built using `lv_obj_class` objects.
- They have custom and complex logic inside.
- They cannot be loaded from XML at runtime because the custom code cannot be loaded.
- They can have a large API with `set/get/add` functions.
- They can themselves contain Widgets as children (e.g. `Tabview`'s tabs, `Dropdown`'s lists).

Any handwritten Widget can be accessed from XML by:

1. Defining its API in an XML file.
2. Writing and registering an XML parser for it. [See some examples here.](#)

## Components

*Components* are built from Widgets and/or other Components, and **can be loaded at runtime**. The main characteristics of Components are:

- In XML, they start with a `<component>` root element.
- They are built in XML only and cannot have custom C code.
- They can be loaded from XML as they don't contain custom C code, only XML.
- They are built from Widgets and/or other Components.
- They can be used for styling Widgets and other Components.

- They can contain (as children) Widgets and/or other Components.
- They can have a simple API to pass properties to their children (e.g. `btn_text` to a Label's text).

Regardless of whether the XML was written manually or by the UI editor, the XML files defining Components can be registered in LVGL, and after that, instances can be created. In other words, LVGL can just read the XML files, "learn" the Components from them, and thereafter create children as part of Screens and other Components.

## Screen

*Screens* are similar to Components:

- In XML, they start with a `<screen>` root element.
- They are built from Widgets and/or other Components to describe the *Screen*.
- They can be loaded from XML at runtime as they describe only visual aspects of the UI.
- They do not have an API.
- They can be referenced in Screen load events.

## Syntax Teaser

Each Widget, Component, or Screen XML file describes a single UI element. The root element for Widgets, Components, and Screens is `<widget>`, `<component>`, and `<screen>` respectively. Other than that, the contained XML elements are almost identical. This is a high-level overview of the most important XML elements that will be children of these root elements:

### `<api>`

Describes the properties that can be `set` for a Widget or Component. Properties can be referenced by `$`. For Widgets, custom enums can also be defined with the `<enumdef>` tag.

### `<consts>`

Specifies constants (local to the Widget or Component) for colors, sizes, and other values. Constant values can be referenced using `#`.

### `<styles>`

Describes style (`lv_style_t`) objects that can be referenced (and shared) by Widgets and Components later.

### `<view>`

Specifies the appearance of the Widget or Component by describing the children and their properties.

This is a simple example illustrating what an LVGL XML Component looks like. Note that only the basic features are shown here.

```
<component>
  <consts>
    <px name="size" value="100"/>
    <color name="orange" value="0xffff00"/>
  </consts>

  <api>
    <prop name="btn_text" default="Apply" type="string"/>
  </api>

  <styles>
    <style name="blue" bg_color="0x0000ff" radius="2"/>
    <style name="red" bg_color="0xff0000"/>
  </styles>

  <view extends="lv_button" width="#size">
    <style name="blue"/>
    <style name="red" selector="pressed"/>
    <my_h3 text="$btn_text"
           color="#orange"
           align="center"/>
  </view>
</component>
```

## Usage Teaser

LVGL's UI editor can be used in two different ways.

### Export C and H Files

The Widgets, Components, Screens, images, fonts, etc., can be converted to .C/.H files having plain LVGL code. The exported code works the same way as if it was written by the user.

In this case, the XML files are not required anymore to run the C code (unless modifications may be made later and code is exported again).

The XML files were used only during editing/implementing the Widgets and Components to save recompilation time and optionally leverage other useful UI Editor features.

### Load the UI from XML

Although the Widgets' code always needs to be exported in C and compiled into the application (just like the built-in LVGL Widgets are also part of the application), the Components' XML can be loaded and any number of instances can be created at runtime.

In the simplest case, a Component can be registered with `lv_xml_component_register_from_file(path)` and an instance can be created with `lv_obj_t * obj = lv_xml_create(parent, "my_button", NULL)`.

Note that loading the UI from XML practically has no impact on performance. Once the XML files are registered and the UI is created, it behaves the same way as if it were created from C code.

Registering XMLs and creating instances is not memory hungry nor slow. The biggest memory overhead is that the `<view>` of the Components is saved in RAM (typically 1–2 kB/component).

## 8.1.2 Component Library

### Panoramica

The collection of Components, Widgets, Screens, Images, Fonts, etc., is called a Component Library.

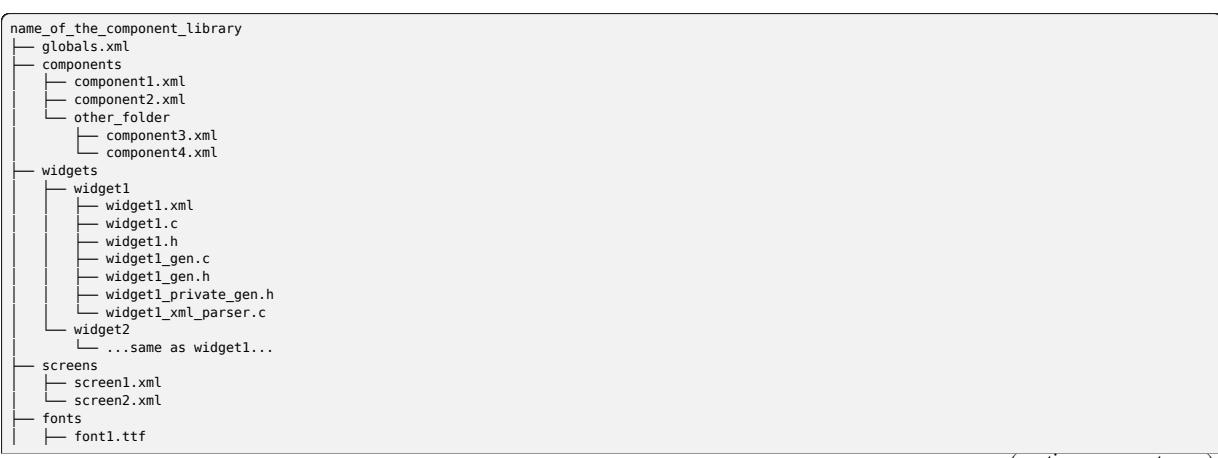
A Component Library can be fully self-sufficient, but it can also reference data from other Component Libraries.

LVGL itself is a Component Library that supplies the built-in Widgets, data types, etc. You can find the XML files that describe the LVGL Widgets [here](#).

A project always has at least 2 Component Libraries: that of LVGL as mentioned above, and its own where the Screens, Components, and Widgets of the project are defined. A project may include additional Component Libraries.

### Structure

A typical structure for a Component Library looks like this:



(continua dalla pagina precedente)



## Visibility

The content of all `globals.xml` files is part of a common global scope, and any Components, Widgets or Screens defined therein can be used in all XML files.

Styles, constants, and other data defined in the XML file of Components, Widgets, or Screens are local to that XML file.

Thus, there are two namespaces:

1. **local namespace** within the given XML file of Components, Widgets and Screens.
2. **global namespace** created from the data in the `globals.xml` file from each Component Library included.

To find names referenced in XML files, the local namespace is checked first, and if a name is not found there, then the global namespace is checked.

The names of defined Components, Widgets and Screens become part of the global namespace and must be unique therein. This ensures that each Component has a unique name.

All data belonging to the LVGL core Component Library is prefixed by `lv_` (e.g., `lv_label`, `lv_font_default`).

A custom Component can be prefixed with `watch_`, `small_`, `light_`, or anything else the developer deems appropriate.

LVGL's UI Editor will show an error if there is a name conflict.

## `globals.xml`

A single `globals.xml` file should be created in the root directory of each Component Library. The definitions in it do not belong to any specific Widget but are available throughout the entire UI, Widgets, and all XML files. The valid tags in it are:

- <config>**  
Can specify name and help.
- <api>**  
Used with `<enumdefs>` to show possible values for Widget or Component attributes.
- <subjects>**  
List of *Subjects*. Can be considered the API of a Component Library.
- <consts>**  
Globally available constants.
- <styles>**  
Globally available styles.
- <fonts>**  
Globally available fonts.
- <images>**  
Globally available images.
- <const\_variants>**  
Vedere di seguito.
- <style\_variants>**  
Vedere di seguito.

The `globals.xml` must be the only `globals.xml` file within the Component Library's folder tree.

From each `globals.xml` file, a `<config.name>.h` file is generated, which is included by all generated header files --- not only in the sub-folders where `globals.xml` is created, but in all exported .C and .H files. This ensures that constants, fonts, and other global data are available for all Widgets and new Widgets.

## Esempio

A `globals.xml` file of a Component Library might look like this:

```
<globals>
  <api>
    <enumdef name="mode">
      <enum name="slow"/>
      <enum name="fast"/>
    </enumdef>
  </api>

  <consts>
    <px name="small_unit" value="8"/>
    <px name="large_unit" value="16"/>
  </consts>

  <styles>
    <style name="style_red" bg_color="0xffff0000">
    </style>
  </styles>

  <images>
    <file name="arrow_left" src="A:/images/arrow_left.png"/>
  </images>

  <fonts>
    <tiny_ttf name="big" src="A:/fonts/arial.ttf" size="28"/>
  </fonts>
</globals>
```

## 8.1.3 Project

### Panoramica

A single `project.xml` file should be created for each project where the following content is specified:

#### <folders>

Specifies the path to Component Libraries. LVGL's base Widgets are always loaded automatically.

#### <targets>

Describes various hardware configurations, allowing the UI Editor to check if the UI is out of resources and to select different previews for each Screen according to the specified displays.

In LVGL's Editor the `folders`, `memory` and ```<display color_format="..." />``` are not supported yet.

## Esempio

```
<project>
  <folders>
    <folder path="../widget_lib1"/>
    <folder path="/home/user/work/ui_libs/modern"/>
    <folder path="https://github.com/user/repo"/>
  </folders>

  <targets>
    <target name="small">
      <display width="320" height="240" color_format="RGB565"/>
      <memory name="int_ram" size="128K"/>
      <memory name="ext_ram" size="2M"/>
      <memory name="int_flash" size="512K"/>
      <memory name="ext_flash" size="32M"/>
    </target>

    <target name="large">
      <display width="1024" height="768" color_format="XRGB8888"/>
      <memory name="int_ram" size="128K"/>
      <memory name="ext_ram" size="2M"/>
      <memory name="int_flash" size="512K"/>
      <memory name="ext_flash" size="32M"/>
    </target>
  </targets>
</project>
```

## 8.1.4 Syntax

### Naming conventions

- A standard XML syntax is used.
- Lowercase letters with `_` separation are used for attribute names.

- Tag names follow the usual variable-name rules: they must start with a letter or '`_`' and the rest of the name may be comprised of letters, '`_`' and digits.
- The LVGL API is followed as much as possible, e.g., `align="center"`, `bg_color="0xff0000"`.
- For colors, all these syntaxes are supported (similar to CSS colors): `0x112233`, `#112233`, `112233`, `0x123`, `#123`, `123`. Note that like CSS, `0x123`, `#123` and `123` all mean `#112233`.
- `params` can be referenced with `$`.
- `consts` can be referenced with `#`.
- `styles` can be attached to states and/or parts like this: `styles="style1 style2:pressed style3:focused:scrollbar"`.
- Local styles (i.e. styles that are stored within the Component and thus not shared by any other Components) can be used like this: `<lv_label style_text_color="0xff0000" style_text_color:checked="0x00ff00"/>`.

## Types

All of the types can be used as API property types, but only a subset of them can be used as constant and *Subject* types.

### Simple types

The following simple built-in types are supported:

<b>bool</b>	a <code>true</code> or <code>false</code> .
<b>int</b>	Integer number in the range of roughly -2B to 2B by default. (Same as <code>int32_t</code> in C.)
<b>px</b>	Simple pixel units. The unit <code>px</code> can be omitted.
<b>%</b>	Percentage. <code>%</code> must be appended to the value as the unit. (Means the same as <code>lv_pct()</code> .)
<b>content</b>	Means <code>LV_SIZE_CONTENT</code> .
<b>string</b>	Simple NUL-terminated string. When multiple strings are used in a property or string array, ' <code>'</code> should be used. E.g. <code>foo=" 'a' 'b' "</code> .
<b>color</b>	A color stored as 24-bit RGB ( <code>lv_color_t</code> ).
<b>opa</b>	Opacity value in the range of 0 to 255 or 0 to 100%. Like CSS, percentage values must be followed by <code>'%</code> .
<b>lv_obj</b>	Pointer to a Widget ( <code>lv_obj_t*</code> ).
<b>screen</b>	Pointer to a screen (also <code>lv_obj_t*</code> ).
<b>time_ms</b>	Means some time in milliseconds unit
<b>deg_0.1</b>	Degrees with 0.1 resolution
<b>scale_1/256</b>	Scale/Zoom, where 256 means 100%, 128 means 50%, 512 means 200% etc.

## Name-based types

In XML files, fonts, images, styles, etc., are not used by pointer but by string names. For example, a style is defined like `<style name="red" bg_color="0xff0000"/>`. Later, they can be referenced by their names.

This means that the actual values need to be bound to the names when the UI is loaded from XML, otherwise, LVGL wouldn't know what a name means.

Most of these connections are done automatically (e.g., for styles, fonts, images, animations, gradients, etc.), but others need to be connected manually (e.g., event callbacks where the callback itself is available only in the code).

For fonts and images, the connections are created automatically if the source is a file. If the font or image is compiled into the application (as a C array), the user needs to specify which variable a given name refers to.

To create these connections, functions like

- `lv_xml_register_image(scope, name, pointer)`
- `lv_xml_register_font(scope, name, pointer)`
- `lv_xml_register_event_cb(scope, name, callback)`
- ecc.

can be used. Later, a pointer to the object can be retrieved by

- `lv_xml_get_image(scope, name)`
- `lv_xml_get_font(scope, name)`
- `lv_xml_get_event_cb(scope, name)`
- ecc.

`scope` can be NULL to use the global scope or `lv_xml_component_get_scope("my_component")` returns the a component's local scope.

### style

Name of a style. `lv_xml_get_style_by_name(&ctx, name)` returns an `lv_style_t*`.

### font

Name of a font. `lv_xml_get_font(&ctx, name)` returns an `lv_font_t*`.

### image

Name of an image. `lv_xml_get_image(&ctx, name)` returns an `const void*`, which can be `lv_image_dsc_t*` or a NUL-terminated string path to a file.

### animation

Name of an animation descriptor. `lv_xml_get_anim(&ctx, name)` returns an `lv_anim_t*`.

### subject

Name of a *Subject*. `lv_xml_get_subject(&ctx, name)` returns an `lv_subject_t*`.

### grad

Name of a gradient. `lv_xml_get_grad(&ctx, name)` returns an `lv_grad_dsc_t*`.

### event\_cb

Name of an event callback. `lv_xml_get_event_cb(&ctx, name)` returns an `lv_event_cb_t`.

### screen\_create\_cb

In XML it's the name of a screen XML file. In exported code it's a function like `lv_obj_t * my_screen_create(void)`

## Arrays

An array of any type can be defined in four ways:

### **int[N]**

An integer array with N elements. In the exported code N is passed a parameter after the array.

### **string[...NULL]**

An array terminated with a NULL element. NULL can be replaced by any value.

### **string[5]**

An array that must have exactly 5 elements. In the exported code only the array will be passed as the Widget assumes the given number of elements.

### **string[]**

No NULL termination and no count parameter, used when the number of elements is not known or delivered via another mechanism, such as via a function parameter.

## Enums

<enumdef> can be used in the <api> tags to create custom enums for **Widgets**. This is not supported for Components.

Ad esempio:

```
<api>
<enumdef name="my_widget_mode" help="Possible modes" help-zh="Chinese help">
  <enum name="normal" help="Normal mode" help-zh="Normal mode in Chinese" value="0x10"/>
  <enum name="inverted" help="Inverted mode"/>
</enumdef>

<prop name="mode" help="help" type="enum:my_widget_mode" help="help"/>
</api>
```

When used as a type, a + suffix means multiple values can be selected and ORed. For example: `type="axis+"`. In this case, the options should be separated by |, for example: `axis=primary_x|secondary_y`.

## Compound types

Types can be compound, meaning multiple options/types are possible. For example, for width: `type="px|%|content"`.

## Limiting accepted values

It is also possible to limit the possible options the user can select from an enum. Ad esempio:

- Enums: `type="dir(top bottom)"`
- Colors: `type="color(0xff0000 0x00ff00 0x0000ff)"`
- Strings: `type="string('Ok' 'Cancel')`

Limiting accepted values is not supported yet, however in the UI Editor if an invalid option is selected, it will be highlighted as an error.

## 8.2 Build UIs

### 8.2.1 Root elements

#### Components

#### Panoramica

Components are one of the main building blocks for creating new UI elements.

<component>s support the following child XML tags:

- <consts>

- `<api>`
- `<styles>`, and
- `<view>`
- `<previews>`

Although they can't contain C code, they are very powerful:

- They can extend another Component or Widget (the base can be defined)
- Components can be built from Widgets and other Components
- A custom API can be defined
- Local styles can be defined, and the global styles can be used
- Local constants can be defined, and the global constants can be used
- Function calls, subject changes, or screen load/create events can be added. See [XML Events](#)
- Previews can be defined to preview the components in various settings

Unlike Widgets (which are always compiled into the application), Components can either:

1. be loaded at runtime from XML, or
2. be exported to C code and compiled with the application.

## Usage from XML

### In XML Files

Using Components in XMLs is very intuitive. The name of the components can be used as XML tag in the `<view>` of other Components, Screens, and Widgets.

The Component properties are just XML attributes.

To load Components from file, it's assumed that the XML files are saved to the device either as data (byte array) or as file. Once the data is saved, each component can be registered, and instances can be created after that.

```
<!-- my_button.xml -->
<component>
  <view extends="lv_button" flex_flow="row">
    <lv_image src="logo"/>
    <my_h3 text="Title"/>
  </view>
</component>
```

*Styles*, *Constants*, and *custom API* can also be described in the XML files.

### Registration

Once a Component is created (e.g., `my_button`), it can be registered by calling either:

- `lv_xml_component_register_from_file("A:lvgl/examples/others/xml/my_button.xml")`
- `lv_xml_component_register_from_data("my_button", xml_data_of_my_button)`

These registration functions process the XML data and store relevant information internally. This is required to make LVGL recognize the Component by name.

Note that the "A:" in the above path is a file system "driver identifier letter" from [File System \(lv\\_fs\\_drv\)](#) and used accordingly. See that documentation for details.

When loaded from a file, the file name is used as the Component name.

During registration, the `<view>` of the Component is saved in RAM.

## Instantiation

After registration, a new instance of any registered Component can be created with:

```
lv_obj_t * obj = lv_xml_create(lv_screen_active(), "my_button", NULL);
```

The created Widget is a normal LVGL Widget that can be used like any other manually-created Widget.

The last parameter can be **NULL** or an attribute list, like this:

```
/* Can be local */
char * my_button_attrs[] = {
    "x", "10",
    "y", "-10",
    "align", "bottom_left",
    NULL, NULL,
};

lv_obj_t * btn1 = lv_xml_create(lv_screen_active(), "my_button", my_button_attrs);
```

## Usage from Exported Code

From each Component XML file, a C and H file is exported with a single function inside:

```
lv_obj_t * component_name_create(lv_obj_t * parent, ...api properties...);
```

where 'component\_name' (in the function above) is replaced by the Component's XML file name.

When a Component is used in another Component's XML code and the code is exported, this **Create** function will be called. This means that Components do not have a detailed set/get API but can be created with a fixed set of parameters.

If the user needs to access or modify values dynamically, it is recommended to use a *Data bindings via Subject*.

The user can also call these **...\_create()** functions at any time from application code to create new components on demand.

## Extending

Additionally, when a Component is created, it can use the extended Widget's attributes (see `<view extends="...">` in the code examples below).

This means that Components inherit the API of the extended Widget as well.

## Custom Properties

The properties of child elements can be adjusted, such as:

```
<my_button x="10" width="200"/>
```

However, it's also possible to define custom properties in the `<api>` tag. The properties then can be passed to any properties of the children by referencing them by \$. Ad esempio:

```
<!-- my_button.xml -->
<component>
  <api>
    <prop name="btn_text" type="string"/>
  </api>

  <view extends="lv_button">
    <lv_label text="$btn_text"/>
  </view>
</component>
```

And it can be used like

```
<!-- my_list.xml -->
<component>
  <view>
    <my_button btn_text="First"/>
    <my_button btn_text="Second"/>
    <my_button btn_text="Third"/>
  </view>
</component>
```

In this setup, the `btn_text` property is mandatory, however it can be made optional by setting a default value:

```
<prop name="btn_text" type="string" default="Title"/>
```

See [API](#) for more details and [XML Syntax](#) for all the supported types.

## Esempi

The following example demonstrates parameter passing and the use of the `text` label property on a Component. Styles and Constants are also shown.

```
<!-- h3.xml -->
<component>
  <view extends="lv_label" style_text_color="0xfffff00"/>
</component>
```

```
<!-- red_button.xml -->
<component>
  <api>
    <prop type="string" name="btn_text" default="None"/>
  </api>

  <consts>
    <int name="thin" value="2"/>
  </consts>

  <styles>
    <style name="pressed_style" border_width="#thin" border_color="0xff0000"/>
  </styles>

  <view extends="lv_button" style_radius="0" style_bg_color="0xff0000">
    <style name="pressed_style" selector="pressed"/>

    <h3 text="Some text"/>
    <h3 text="$btn_text" y="40"/>
  </view>
</component>
```

```
lv_xml_component_register_from_file("A:path/to/h3.xml");
lv_xml_component_register_from_file("A:path/to/red_button.xml");

/* Creates a button with "None" text */
lv_xml_create(lv_screen_active(), "red_button", NULL);

/* Use attributes to set the button text */
const char * attrs[] = {
  "btn_text", "Click here",
  NULL, NULL,
};
lv_xml_create(lv_screen_active(), "red_button", attrs);
```

## Live Example

### Load components at runtime

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_XML

void lv_example_xml_1(void)
{
  /*A red button created from built-in LVGL widgets
   *It has an API parameter too to change its text.*/
  const char * red_button_xml =
  "<component>
    <api>
      <prop name=\"button_text\" type=\"string\" default=\"None\"/>
    </api>
    <view extends=\"lv_button\" radius=\"0\" style_bg_color=\"0xa91500\">
      <lv_label text=\"$button_text\" align=\"center\"/>
    </view>
  </component>";

  /*The card is just an lv_obj where a label and two red buttons are used.
   * Its API allow setting a title (label test) and the action (the text of a button)*/
  const char * card_xml =
  "<component>
    <api>
      <prop name=\"title\" type=\"string\" default=\"Hello world\"/>
      <prop name=\"action\" type=\"string\"/>
    </api>
    <view width=\"200\" height=\"content\">
      <lv_label text=\"$title\" align=\"top_mid\"/>
      <red_button y=\"20\" align=\"top_left\" button_text=\"Cancel\"/>
      <red_button y=\"20\" align=\"top_right\" button_text=\"$action\"/>
    </view>
  </component>";

  /* Motor card is a special case of a card where the title and action are already set*/
  const char * motor_card_xml =
  "<component>
    <view extends=\"card\" title=\"Motor start?\" action=\"Start\"/>
  </component>";

  /*Register all the custom components*/
  lv_xml_component_register_from_data("red_button", red_button_xml);
  lv_xml_component_register_from_data("card", card_xml);
  lv_xml_component_register_from_data("motor_card", motor_card_xml);
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * card;
/*Create a card with the default values/
card = (lv_obj_t *) lv_xml_create(lv_screen_active(), "card", NULL);

/*Create a motor card too. The returned value can be adjusted freely*/
card = (lv_obj_t *) lv_xml_create(lv_screen_active(), "motor_card", NULL);
lv_obj_set_y(card, 90);

/*Pass properties to a card*/
const char * attrs[] = {
    "y", "180",
    "action", "Apply",
    "title", "New title",
    NULL, NULL,
};
card = (lv_obj_t *) lv_xml_create(lv_screen_active(), "card", attrs);

}
#endif

```

```

#include "../../../lv_examples.h"
#if LV_BUILD_EXAMPLES && LV_USE_XML && LV_USE_TRANSLATION

void lv_example_xml_2(void)
{
    lv_result_res;
    res = lv_xml_component_register_from_file("A:lvgl/examples/others/xml/my_h3.xml");
    if(res != LV_RESULT_OK) {
        lv_obj_t * label = lv_label_create(lv_screen_active());
        lv_label_set_text(label, "Couldn't open the XML files.");
        lv_obj_center(label);
        return;
    }
    lv_xml_component_register_from_file("A:lvgl/examples/others/xml/my_card.xml");
    lv_xml_component_register_from_file("A:lvgl/examples/others/xml/my_button.xml");
    lv_xml_component_register_from_file("A:lvgl/examples/others/xml/view.xml");
    lv_xml_translation_register_from_file("A:lvgl/examples/others/xml/translations.xml");

    lv_xml_register_font(NULL, "lv_montserrat_18", &lv_font_montserrat_18);

    lv_translation_set_language("de");

    lv_obj_t * obj = (lv_obj_t *) lv_xml_create(lv_screen_active(), "view", NULL);
    lv_obj_set_pos(obj, 10, 10);

    lv_xml_component_unregister("my_button");

    const char * sliderAttrs[] = {
        "x", "200",
        "y", "-15",
        "align", "bottom_left",
        "value", "30",
        NULL, NULL,
    };

    lv_obj_t * slider = (lv_obj_t *) lv_xml_create(lv_screen_active(), "lv_slider", sliderAttrs);
    lv_obj_set_width(slider, 100);
}
#endif

```

## API

### I Widget

#### Panoramica

Besides components and Screens, Widgets are the other main building blocks of UIs. The XML file of the Widgets is wrapped in a `<widget>` XML root element.

`<widget>`s support the following child XML tags:

- `<consts>`
- `<api>`
- `<styles>`, and
- `<view>`
- `<previews>`

Just like Components, Widgets also can be the children of other Widgets and Components.

The main difference is that Widgets are written in C and compiled into the application. It means unlike Components, Widgets can contain custom C code. For example, when a property is set, any complex C code can run to set that value.

To connect the C code to XML, XML parser functions need to be implemented and registered. It's pretty simple as LVGL already provides all the helper functions and the required libraries. Also there are many [XML parser examples](#) here.

## Built-in Widgets

The built-in LVGL widgets (e.g. *Slider* (`lv_slider`), *Label* (`lv_label`), *Chart* (`lv_chart`), etc.) already have XML parsers and therefore are available in XML.

Ad esempio:

```
<view>
<lv_label x="10" text="Hello"/>
</view>
```

The built-in widgets are

- pure C code (e.g. `lv_slider.c`)
- an XML file to define only the API (e.g. `lv_slider.xml`). It is used only in the UI Editor to validate and autocomplete properties.
- an XML parser C file to map the XML attributes to C functions. (e.g. `lv_xml_slider_parser.c`)

## Creating New Widgets

It's possible to create new widgets in the same way as the built-in LVGL widgets are handled.

However, using the UI Editor it's much faster and simpler. When an XML file is created and the `<widget>` root element is used, the following .C/.H files are generated automatically:

### `<widget_name>.gen.h`

Contains the generated API implementation of the widget (overwritten on each code export).

### `<widget_name>.private_gen.h`

Contains private API and the data for the widget (overwritten on each code export).

### `<widget_name>.gen.c`

Contains the internals of the Widget, e.g. constructor with the children, destructors, event handler, etc. (overwritten on each code export).

### `<widget_name>.h`

Includes `<widget_name>.gen.h` and allows the user to define custom APIs. Only a skeleton is exported once.

### `<widget_name>.c`

Contains hooks from `<widget_name>.gen.c` and allows the user to write custom code. Only a skeleton is exported once.

### `<widget_name>.xml_parser.c`

Processes the XML strings and calls the required functions according to the set attributes. Only a skeleton is exported once.

## Utilizzo

### XML Parser

To make the Widgets accessible from XML, an XML parser needs to be created and registered for each Widget. The XML parser for the label Widget looks like this:

```
void * lv_xml_label_create(lv_xml_parser_state_t * state, const char ** attrs)
{
    /* Create the label */
    void * obj = lv_label_create(lv_xml_state_get_parent(state));
    return obj;
}

void lv_xml_label_apply(lv_xml_parser_state_t * state, const char ** attrs)
{
    void * obj = lv_xml_state_get_item(state);

    /* Apply the common properties, e.g., width, height, styles, flags, etc. */
    lv_xml_obj_apply(state, attrs);

    /* Process the label-specific attributes */
    for(int i = 0; attrs[i]; i += 2) {
        const char * name = attrs[i];
        const char * value = attrs[i + 1];
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

    if(lv_streq("text", name)) lv_label_set_text(obj, value);
    if(lv_streq("long_mode", name)) lv_label_set_long_mode(obj, long_mode_text_to_enum(value));
    /* Process more props here */
}

/* Helper to convert strings to enum values */
static lv_label_long_mode_t long_mode_text_to_enum(const char * txt)
{
    if(lv_streq("wrap", txt)) return LV_LABEL_LONG_WRAP;
    if(lv_streq("scroll", txt)) return LV_LABEL_LONG_SCROLL;

    LV_LOG_WARN("%s is an unknown value for label's long_mode", txt);
    return 0; /* Return 0 in the absence of a better option. */
}

```

A Widget XML processor can be registered like this:

```
lv_xml_widget_register("lv_label", lv_xml_label_create, lv_xml_label_apply);
```

After registration, a Widget can be created like this from C code:

```

const char * attrs[] = {
    "text", "Click here",
    "align", "center",
    NULL, NULL,
};

lv_xml_create(lv_screen_active(), "lv_label", attrs);

```

And in XML it can be used like

```
<view>
<lv_label width="100" text="I'm a label!" wrap="scroll"/>
</view>
```

## Adding Custom Code

`<widget_name>.c` contains three hooks:

- **Constructor hook:** Called when the Widget and all its children are created. Any modifications can be done on the children here.
- **Destructor hook:** Called when the Widget is deleted. All manually allocated memory needs to be freed here.
- **Event hook:** Called at the beginning of the Widget's event callback to perform any custom action.

In this C file, the `set` functions for each API `<prop>` also need to be implemented. The declaration of these functions is automatically exported in `<widget_name>_gen.h`.

Besides these, any custom code and functions can be freely implemented in this file.

## Elements

Elements are internal parts of the widget that can be accessed and/or created dynamically. For example, tabs of a tabview, list of a dropdown, series of a chart, etc.

Just like any other Widget API properties, Elements also can be defined in the `<api>` tag of the Widget's XML.

Learn more about the Elements in the documentation page of [`<api>`](#).

## Screen

### Panoramica

Screens work very similarly to Components. Both can be:

- Loaded from XML
- Contain Widgets and Components as children

Screens are wrapped in the `<screen>` XML root element and used to organize the content of the UI.

Screens can have only the following child XML tags:

- `<consts>`

- `<styles>`, and
- `<view>`

That is, Screens **cannot** have an `<api>` or `<preview>`.

## Utilizzo

Each XML file describes a *Screen*. The name of the XML file will also be the name of the Screen.

This example illustrates a screen in XML. In the example, a `<my_header>` and a `<my_main_cont>` component is used to keep the screen simple.

```
<screen>
  <consts>
    <string name="title" value="Main menu"/>
  </consts>

  <styles>
    <style name="dark" bg_color="#0x333"/>
  </styles>

  <view>
    <my_header label="#title"/>
    <my_main_cont>
      <style name="dark"/>
      <button text="Weather" icon="cloudy"/>
      <button text="Messages" icon="envelope"/>
      <button text="Settings" icon="cogwheel"/>
      <button text="About" icon="questionmark"/>
    </my_main_cont>
  </view>
</screen>
```

## Code export

When the C code is exported from the UI Editor, `screen_name_gen.c/h` files are exported, containing only a single `lv_obj_t * screen_name_create(void)` create function.

By using this function, any number of screen instances can be created and loaded as needed.

## Preview

Screens don't support the `<preview>` tag because it doesn't make sense to preview each screen in different resolutions.

As Screens are related to the target hardware in the `project.xml` file, multiple `<display>` elements can be defined. In the UI Editor, when a Screen is being developed, the user can select from all the defined displays in the Preview, and the Screen will be shown with the given resolution and color depth.

This is useful for verifying responsive designs.

## Eventi

It's very common to load or create Screens on a button click or other events.

Both are supported by adding special XML tags as children of Components or Widgets:

```
<view>
  <lv_button>
    <lv_label text="Click or Long press me"/>
    <!-- Load an already created screen that has the name "first".
        Note that here the name of the instance is used,
        and not the name of the XML file. -->
    <screen_load_event screen="first" trigger="clicked" anim_type="fade"/>
    <!-- Create an instance of "about" and load it.
        Note that here the name of the XML file is used. -->
    <screen_create_event screen="about" trigger="long_pressed"/>
  </lv_button>
</view>
```

Learn more on [XML Events](#) page.

## 8.2.2 Main tags

### API

The `<api>` tag can be a child of `<widget>` and `<component>` tags. (`<screen>`s don't support custom APIs.)

The only common point is that both Widgets and Components support having `<prop>` (properties) in the `<api>` tag to describe their interface.

However, as Widgets and Components work very differently (Widgets have C code, but Components are pure XML), even properties are interpreted differently.

### Components

#### Panoramica

While Widgets can have complex `set/get` APIs, Components are very simple.

When their XML is converted to a C file, only a `create` function is generated, where all the `<prop>`s are arguments. Ad esempio:

```
<api>
<prop name="prop1" type="int"/>
<prop name="prop2" type="string"/>
</api>
```

This generates the following C function:

```
lv_obj_t * my_component_create(lv_obj_t * parent, int32_t prop1, const char * prop2);
```

These properties are set once (at creation time), and there are no specific `set` functions to modify the property later. LVGL's general API can still be used to modify any widget in the component, but no dedicated API functions are generated.

#### Referencing properties

`<prop>`s are simply forwarded to widget or component APIs. For example, if a component has `<prop name="button_label" type="string"/>`, it can be used in a label as `<lv_label text="$button_label"/>`.

In the generated code, these are passed as arguments in `create/set` functions.

#### Default values

Since each property is passed as an argument to the `create` function, each must have a value. This can be ensured by:

- Simply setting them in the XML instance
- Providing a default value in the `<api>`, e.g., `<prop name="foo" type="string" default="bar"/>`

#### Limitazioni

Note that none of the Widget API features such as `<param>`, `<enumdef>`, or `<element>` can be used for Components. Only simple properties that are forwarded are supported.

#### Esempio

```
<!-- my_button.xml -->
<component>
<api>
    <prop name="button_icon" type="image" default="NULL"/>
    <prop name="button_label" type="string" default="Label"/>
</api>
<view extends="lv_button" flex_flow="row" width="100%">
    <lv_image src="$button_icon" inner_align="stretch" width="16" height="16"/>
    <lv_label text="$button_label"/>
</view>
```

(continues on next page)

(continua dalla pagina precedente)

```
</component>
<!-- my_list.xml -->
<component>
    <view flex_flow="column">
        <my_button button_label="First"/>
        <my_button button_label="Wifi" button_icon="img_wifi"/>
        <my_button button_label="Third"/>
    </view>
</component>
```

## I Widget

### Properties

Properties are the core part of describing a Widget's API.

```
<api>
    <prop name="text" type="string" help="Text of the label."/>
</api>
```

### Parameters

Some properties take multiple parameters. For example: `lv_label_set_bind_text(label, subject, "%d °C")`

It's described as:

```
<api>
    <prop name="bind_text" help="Bind a subject's value to a label.">
        <param name="bind_text" type="subject" help="Integer or string subject"/>
        <param name="fmt" type="string" help="Format string, e.g. %d °C "/>
    </prop>
</api>
```

And used as:

```
<lv_label bind_text="subject" bind_text-fmt="%d °C"/>
```

Parameters with the same name as the property can be referenced directly. Other parameters use `property-param` notation.

Unset parameters fall back to:

- Their default value (if defined)
- Type-specific defaults (e.g., 0, false, NULL)

### Mapping

Each `<prop>` is mapped to a `set` function. This mapping is implemented in the Widget's XML parser. See [the LVGL XML parsers](#).

If `<param>`'s are used, they are passed to the same ```set` function. If a property is not set on a Widget instance, it is skipped and the Widget's built-in default is used.

### `<enumdef>`

Only used with Widgets, this tag defines enums for parameter values.

```
<api>
    <enumdef name="my_widget_mode" help="Possible modes">
        <enum name="normal" help="Normal mode" value="0x10"/>
        <enum name="inverted" help="Inverted mode"/>
    </enumdef>
    <prop name="mode" help="Set Widget mode">
        <param name="mode" type="enum:my_widget_mode"/>
    </prop>
</api>
```

Enum values are ignored in export; the names are used and resolved by the compiler. XML parsers must handle mapping enum names to C enums.

## <element>

Also exclusive to Widgets, elements define sub-widgets or internal structures (e.g., chart series, dropdown list, tab views).

They support <arg> and <prop>:

- <arg>s are required and used when creating/getting the element.
- <prop>s are optional and mapped to setters.

Elements are referenced as <widget-element> in views.

Name parts are separated by - (as - is not allowed inside names).

Element access types:

- **add**: Create multiple elements dynamically.
- **get**: Access implicitly created elements.
- **set**: Access indexed parts (e.g., table cells).
- **custom**: Map custom C function to XML.

As **add** and **get** elements return an object they also have a type. This type can be any custom type, for example, **type="my\_data"**. In the exported code the return value will be saved in a **my\_data\_t \* variable**.

If the type is **type="lv\_obj"** it allows the element to have children widgets or components.

Note that, only the API can be defined in XML for elements; implementations must be in C.

### access="add"

Elements are created via an **add** function:

```
<api>
  <element name="indicator" type="obj" help="The indicator of my_widget" access="add">
    <arg name="color" type="color"/>
    <arg name="max_value" type="int"/>
    <prop name="value">
      <param name="value" type="int"/>
    </prop>
  </element>
</api>
```

Used in a view:

```
<my_widget width="100px">
  <my_widget-indicator name="indic1" color="0xff0000" max_value="120" value="30"/>
</my_widget>
```

Generates:

```
lv_obj_t * my_widget_add_indicator(lv_obj_t * parent, lv_color_t color, int32_t max_value);
void my_widget_set_indicator_value(lv_obj_t * obj, int32_t value);
```

### access="get"

Used for internal/implicit elements:

```
<api>
  <element name="control_button" type="obj" help="A control button of my_widget" access="get">
    <arg name="index" type="int"/>
    <prop name="title" type="string"/>
  </element>
</api>
```

Used in a view:

```
<my_widget width="100px">
  <my_widget-control_button name="btn1" index="3" title="Hello"/>
</my_widget>
```

Generates:

```
lv_obj_t * my_widget_get_control_button(lv_obj_t * parent, int32_t index);
void my_widget_set_control_button_title(lv_obj_t * obj, const char * text);
```

**access="set"**

Used for indexed access, like setting values in a table:

```
<api>
  <element name="item" type="obj" access="set">
    <arg name="index" type="int"/>
    <prop name="icon" type="img_src"/>
    <prop name="color" type="color"/>
  </element>
</api>
```

Used in a view:

```
<my_widget width="100px">
  <my_widget-item index="3" icon_src="image1" color="0xffff0000"/>
</my_widget>
```

Generates:

```
void my_widget_set_item_icon(lv_obj_t * parent, int32_t index, const void * icon_src);
void my_widget_set_item_color(lv_obj_t * parent, int32_t index, lv_color_t color);
```

**access="custom"**

Used to describe any custom API functions with a custom name. "custom" elements can have only arguments and no **type** so they are pure setters.

```
<element name="bind_color" access="custom">
  <arg name="subject" type="subject"/>
  <arg name="new_color" type="color"/>
  <arg name="ref_value" type="int"/>
</element>
```

Used in a view:

```
<my_widget width="100px">
  <my_widget-bind_color subject="subject_1" color="0xffff0000" ref_value="15"/>
</my_widget>
```

Generates:

```
void my_widget_bind_color(lv_obj_t * parent, lv_subject_t * subject, lv_color_t color, int32_t ref_value);
```

**Constants****Panoramica**

Constants can be defined to replace any value with a selected type or to be used as special values.

The supported types are: - **color** - **px** - **percentage** - **string** - **opa** - **bool**

**Utilizzo**

```
<consts>
  <color name="color1" value="0xffff0000" help="Primary color"/>
  <px name="pad_xs" value="8" help="Small padding"/>
</consts>
```

Constants can be used in:

- Style properties
- Widget and Component properties

And they can be used like this:

```
<styles>
  <style name="style1" bg_color="#color1"/>
</styles>
```

## Stili

### Panoramica

In XML files, both style sheets (`lv_style_t`) and local styles can be used.

### Style Sheets

In the `<styles>` section, styles and their properties can be defined like this:

```
<style name="red"
      help="What is this style about?"
      border_width="2px"
      border_color="#00ff0000"/>
```

Styles can be referenced like this in the `<view>`:

```
<view>
  <lv_slider value="20">
    <style name="main"/>
    <style name="red" selector="knob"/>
    <style name="blue" selector="knob focused"/>
  </lv_slider>
</view>
```

As shown in the example, parts and states can be set as `selector`.

### Stili Locali

Local styles can be used directly in a Widget, for example:

```
<lv_label style_bg_opa="200" style_bg_opa:disabled="100"/>
```

### Gradients

Before the `<styles>` tag, the `<gradients>` tag can be used to describe various gradients, which can later be referenced in styles.

When a gradient is created, it can be referenced by its name, like:

```
<style bg_grad="grad1"/>
```

O

```
<lv_button style_bg_grad="grad1"/>
```

### Horizontal or Vertical Gradient

To define a simple `<horizontal>` or `<vertical>` gradients:

```
<gradients>
  <horizontal name="grad1">
    <stop color="#ff0000" offset="20%" opa="40%"/>
    <stop color="#00ff00" offset="128" opa="100%"/>
  </horizontal>
</gradients>
```

### Linear Gradient

To define a skewed gradient from two points:

```
<gradients>
  <linear name="grad1" start="50 50" end="100 80">
    <stop color="#ff0000" offset="20%" opa="100%"/>
    <stop color="#00ff00" offset="240" opa="100%"/>
  </linear>
</gradients>
```

## Gradiente Radiale

To define a radial gradient:

```
<gradients>
  <radial name="grad1" center="100 50%" edge="200 50" focal_center="50 80%" focal_edge="55 80">
    <stop color="#ff0000" opa="100%"/>
    <stop color="#00ff00" opa="100%"/>
  </radial>
</gradients>
```

## Gradiente Conico

To define a conical gradient:

```
<gradients>
  <conical name="grad1" center="80 50%" angle="45 270">
    <stop color="#ff0000" opa="100%"/>
    <stop color="#00ff00" opa="100%"/>
  </conical>
</gradients>
```

## View

### Panoramica

The `<view>` tag can be used in:

- `<component>`s
- `<widget>`s
- `<screen>`s
- `<test>`s

to describe how these items look. Inside `<view>`, children can be added in a nested way using already defined `widgets` and `components`. Ad esempio:

```
<view>
  <lv_button width="200">
    <my_icon src="image1"/>
    <lv_label text="Click me"/>
  </lv_button>
</view>
```

## Extends

The `<view>` itself is also a widget or component, which will become the parent of the children. To define the type of the `<view>`, use the `extends` attribute. For example, `extends="lv_slider"`. In this case, an `lv_slider` will be created first, and the children will be added to it.

By adding properties to the `<view>`, the extended type can be customized. Ad esempio:

```
<view extends="lv_slider" width="100%" style_bg_color="0xff8800" flex_flow="row">
  <style name="my_style" selector="pressed[knob]"/>
  <lv_label text="Current value: ">
  <lv_label bind_text="subject_1"/>
</view>
```

Rules for the allowed values of `extends`:

- `<component>`: can extend both `<widget>`s and `<component>`s
- `<widget>`: can extend only `<widget>`s
- `<screen>`: cannot extend anything
- `<test>`: can extend `<widget>`, `<component>`, or `<screen>`

## Animazioni

### Panoramica

XML animations are built on top of *Timeline animations*.

Timelines are composed of simple animations. For example: "change the ``bg\_opa`` of ``my\_button\_2`` from 0 to 255 in 500 ms."

Each component can define its own timeline animations, which can then be played by the component itself or by any parent components.

### Defining Timelines

Timelines can be defined inside `<screen>`s and `<component>`s. For `<widget>`s, timelines are supported only in LVGL's UI Editor, where C code can also be exported from them.

Esempio:

```
<animations>
    <!-- Show the component and its children -->
    <timeline name="load">
        <animation prop="translate_x" target="self" start="-30" end="0" duration="500"/>
        <animation prop="opa" target="icon" start="0" end="255" duration="500"/>
        <animation prop="opa" target="text" start="0" end="255" duration="500" delay="200"/>
    </timeline>

    <!-- Shake horizontally -->
    <timeline name="shake">
        <animation prop="translate_x" target="self" start="0" end="-30" duration="150"/>
        <animation prop="translate_x" target="self" start="-30" end="30" duration="300" delay="150"/>
        <animation prop="translate_x" target="self" start="30" end="0" duration="150" delay="450"/>
    </timeline>
</animations>

<view>
    <lv_button width="200">
        <my_icon name="icon" src="image1"/>
        <lv_label name="text" text="Click me"/>
    </lv_button>
</view>
```

In summary: inside `<animations>`, you can define `<timeline>`s, each with a unique name that you can reference later.

Inside a `<timeline>`, you add `<animation>`s to describe each step. Supported properties of `<animation>` are:

- **prop**: Style property to animate. All integer style properties are supported (colors are not).
- **selector**: Style selector, e.g. `knob|pressed`. Default: `main|default`.
- **target**: Name of the UI element to animate. `self` refers to the root element of the component (the `<view>`).
- **start**: Start value (integer only).
- **end**: End value (integer only).
- **duration**: Duration of the animation in milliseconds.
- **delay**: Delay before starting in milliseconds.
- **early\_apply**: If `true`, the start value is applied immediately, even during the delay.

### Playing Timelines

Timelines can be triggered by events (e.g. click) using `<play_timeline_event>` as a child of any widget.

Esempio:

```
<view>
    <lv_label name="title" text="Hello world!"/>
    <custom_button name="button" y="20">
        <play_timeline_event trigger="clicked" target="button" timeline="bounce"/>
        <lv_label text="Click me"/>
    </custom_button>
</view>
```

You set a `target` UI element and select one of its `timeline`'s to play. If ```target="self"`'', the timeline is looked up in the current component/widget/screen (i.e. in the current XML file).

You can also set `delay` and `reverse="true"` when playing a timeline.

## Under the Hood

Understanding how timelines work internally helps use them effectively.

When an XML file is registered, the contents of the `<animations>` section are parsed, and the animation data is stored as a blueprint.

When an instance of a component or screen is created, `lv_anim_timelines` are created and initialized from the saved blueprint. Each instance gets its own copy.

When a `<play_timeline_event>` is added to a UI element, the target and timeline names are saved as strings. (It can't use pointers as the event can reference UI elements that will be created only later in the `<view>`.)

Finally, when the trigger event happens, LVGL finds the target widget by the saved name, retrieves the specified timeline, and starts it.

Since each instance has its own timeline, you can have multiple components (e.g. 10 `<list_item>`s) and play their `load` timelines independently with different delays.

## Preview

### Panoramica

In `<component>` and `<widget>`, it is possible to define `<preview>` tags. These are **not** exported to code and are **not** loaded from XML.

They are used only by the UI Editor to describe the context of the Component. For example, you might want to:

- Change the background of the Editor's preview to dark.
- Center the Component.
- Set margins.
- Change the size of the preview.

`style_radius` can be used to make preview rounded.

## Utilizzo

You can think of a `<preview>` tag as an `lv_obj` where the following properties can be used:

- `width, height`
- Any local style properties, for example, `style_bg_color="0x333"`
- `flex` and `flex_flow`

It is also possible to define multiple previews, and in the UI Editor, you can select one of them.

## Esempio

```
<component>
  <previews>
    <preview name="small_dark_round"
      width="240" height="240"
      style_bg_color="0x333" style_pad_all="32"
      style_radius="32"/>
    <preview name="large_light"
      width="1980" height="1080"
      style_bg_color="0xeeeeee"/>
  </previews>
  <view>
    ...
  </view>
```

(continues on next page)

(continua dalla pagina precedente)

&lt;/component&gt;

## 8.2.3 Assets

### Immagini

#### Panoramica

An `<images>` section can be added to `globals.xml` files. If present, it describes how to map images with names.

#### Utilizzo

```
<images>
  <file name="avatar" src_path="avatar1.png"/>
  <data name="logo" src_path="logo1.png" color-format="rgb565" />
</images>
```

- `<file>` means that the image source is used as a file path:
- `<data>` means that the image is converted to a C array on export.

In both cases in the exported C code global `const void * <image_name>` variables are created and in the initialization function of the Component Library (e.g. `my_lib_init_gen()`) either the path or the pointer to the converted `lv_image_dsc_t` pointers are assigned to that variable.

In `lv_image_set_src(image, image_name)` `image_name` is used instead of the path or `lv_image_dsc_t` pointer.

### Registering images

If the UI is created from XML at runtime and a `globals.xml` is parsed, the `<data>` tags are skipped because it is assumed that the user manually created the mapping. This is because the XML parser cannot automatically map an image like:

```
lv_image_dsc_t my_logo;
```

to

```
<data name="my_logo"/>
```

To register an image path or data in the XML engine use:

```
lv_xml_register_image(scope, "image_name", data)
```

`scope` is usually `NULL` to register the image in the global scope. To register an image locally for a component you can get its scope with:

```
lv_xml_component_get_scope("component_name")
```

After calling this function, when "`image_name`" is used as an image source in XML, `data` (can be a path or a pointer to an image descriptor) will be used.

### Notes for the UI Editor

For simplicity, in the UI editor's preview, images are always loaded as files. It makes the preview dynamic so no code export and compilation is needed when an image changes.

## Font

### Panoramica

A `<fonts>` section can be added in `globals.xml` files. Later, it might be supported in Components and Widgets to define local fonts and keep the global space cleaner.

### Utilizzo

The following section creates a mapping between font names and their paths with various attributes:

```
<font>
<bin as_file="false" name="medium" src="path/to/file.ttf" range="0x20-0x7f" symbols="" size="24"/>
<tiny_ttf as_file="true" name="big" src_path="path/to/file.ttf" range="0x20-0x7f" symbols="auto" size="48"/>
<freetype name="chinese" src_path="file.ttf" size="48" custom_freetype_attribute="abc"/>
</font>
```

In `<styles>` and `<view>`, fonts can then be referenced by their name, e.g.,

```
<style name="style1" text_font="medium"/>
```

The tag name determines how the font is loaded. Currently, only `tiny_ttf as_file="true"` is supported.

- `bin`:
  - If `as_file="true"`: Converts the font file to `bin` (see `lv_font_conv`) which will be loaded by `lv_binfont_create()`.
  - If `as_file="false"` (default): On export, the font file will be converted to a C array LVGL font that can be used directly by LVGL.
- `tiny_ttf`:
  - If `as_file="true"`: Can be loaded directly by `lv_tiny_ttf_create_file()`.
  - If `as_file="false"` (default): The font file will be converted to a raw C array on export that will be loaded by `lv_tiny_ttf_create_data()`.
- `freetype`: The file can be loaded directly by `lv_freetype_font_create()`.

For simplicity, if `as_file="false"`, fonts will be loaded as files in the preview. Setting `as_file="false"` affects only the C export.

If the UI is created from XML at runtime and a `globals.xml` is parsed, the `as_file="false"` tags are skipped because it is assumed that the user manually creates the mapping. This is because the XML parser cannot automatically map an LVGL font definition like:

```
lv_font_t my_font_24;
```

to

```
<bin name="my_font_24"/>
```

### Exported Code

When C code is exported, global `const lv_font_t * <font_name>` variables are created, and in the initialization function of the Component Library (e.g., `my_lib_init_gen()`), the actual font is assigned.

In `lv_style_set_text_font(&style1, <font_name>)`, the created font is referenced.

### Default Font

"`lv_font_default`" can be used to access `LV_FONT_DEFAULT`. Other built-in fonts are not exposed by default.

## Registering fonts

If the UI is created from XML at runtime and a `globals.xml` is parsed, the `<... as_file="false">` tags are skipped because it is assumed that the user manually created the mapping. This is because the XML parser cannot automatically map fonts like:

```
lv_font_t my_font;
```

to

```
<data name="my_font"/>
```

To register a font in the XML engine use:

```
lv_xml_register_font(scope, "font_name", &my_font);
```

`scope` is usually `NULL` to register the font in the global scope. To register a font locally for a component you can get its scope with:

```
lv_xml_component_get_scope("component_name");
```

After calling this function, when "`font_name`" is used as a font in XML, `&my_font` will be used.

## Notes for the UI Editor

For simplicity, in the UI editor's preview, fonts are always loaded as tinytt fonts. It makes the preview dynamic as no code export and compilation is needed when a font changes.

### 8.2.4 Eventi

#### Panoramica

There are several ways to define events for user interactions. These events can be added as children of any widget.

#### Triggers

In all event types, the `trigger` attribute defines what kind of user action should trigger the event. All LVGL event types are supported with straightforward mapping:

- `LV_EVENT_ALL`: "all"
- `LV_EVENT_CLICKED`: "clicked"
- `LV_EVENT_PRESSED`: "pressed"
- ecc.

#### Call function

User-defined functions can be called like this:

```
<view>
  <lv_button width="200" height="100">
    <event_cb callback="my_callback_1" trigger="clicked" user_data="some_text"/>
    <lv_label text="Hello"/>
  </lv_button>
</view>
```

When the XML is loaded at runtime, the callback name needs to be mapped to a function using `lv_xml_register_event_cb("my_callback_1", an_event_handler)`.

The callback should follow the standard LVGL event callback signature: `void an_event_handler(lv_event_t * e);`

In the exported C code, it is assumed that a function with the exact name exists. For example, `callback="my_callback_1"` will be exported as:

```
void my_callback_1(lv_event_t * e); /* At the beginning of the exported file */
lv_obj_add_event_cb(obj, my_callback_1, LV_EVENT_CLICKED, "some_text");
```

The `user_data` is optional. If omitted, `NULL` will be passed.

## Screen Load and Create events

By using the `<screen_load_event>` and `<screen_create_event>` tags as a child of a widget or component, screens can be loaded or created on a trigger (e.g. click).

The difference between load and create is that:

- **load**: Just loads an already existing screen. After leaving the screen, it remains in memory, so all states are preserved.
- **create**: The screen is created dynamically, and when leaving the screen, it is deleted, so all changes are lost (unless they are saved in `subjects`).

Both tags support the following optional attributes:

- **trigger**: Event code that triggers the action (e.g. `"clicked"`, `"long_pressed"`, etc). Default: `"clicked"`.
- **anim\_type**: Describes how the screen is loaded (e.g. `"move_right"`, `"fade_in"`). Default: `"none"`.
- **duration**: Length of the animation in milliseconds. Default: 0. Only used if `anim_type` is not `"none"`.
- **delay**: Wait time before loading the screen in milliseconds. Default: 0.

This is a simple example of both load and create:

```
<!-- screen1.xml -->
<screen>
  <view style_bg_color="0xff7788">
    <lv_button>
      <lv_label text="Create"/>
      <!-- Create an instance of "screen2" and load it. -->
      <screen_create_event screen="screen2" anim_type="over_right" duration="500" delay="1000"/>
    </lv_button>
  </view>
</screen>

<!-- screen2.xml -->
<screen>
  <view style_bg_color="0x7fff88">
    <lv_button>
      <lv_label text="Load"/>
      <!-- Load an already created instance of screen1.-->
      <screen_load_event screen="screen1"/>
    </lv_button>
  </view>
</screen>
```

```
lv_xml_component_register_from_data("screen1", screen1_xml);
lv_xml_component_register_from_data("screen2", screen2_xml);

/*Create an instance of screen_1 so that it can be loaded from screen2.*/
lv_obj_t * screen1 = lv_xml_create(NULL, "screen1", NULL);
lv_screen_load(screen1);
```

## Set subject value

It's possible to set a `Subject` value on user interaction by adding a special child to any widget:

```
<view>
  <lv_button width="200" height="100">
    <subject_set_int_event trigger="clicked" subject="subject_int" value="10"/>
    <subject_set_float_event trigger="clicked" subject="subject_float" value="12.34"/>
    <subject_set_string_event trigger="clicked" subject="subject_string" value="Hello"/>
    <lv_label text="Set the values"/>
  </lv_button>
</view>
```

The usage is straightforward: the specified `subject` will be set to the given `value` when the `trigger` occurs.

## Increment subject value

Incrementing or decrementing a `Subject` value can be defined as follows:

```
<view>
  <lv_button width="200" height="100">
    <subject_increment_event trigger="clicked" subject="subject_int1" step="10"/>
    <subject_increment_event trigger="clicked" subject="subject_int2" step="-10" min="0" max="50"/>
    <subject_increment_event trigger="clicked" subject="subject_float1" step="2"/>
  </lv_button>
</view>
```

The `<subject_increment_event>` element defines a `step` to be added to the subject's current value when the `trigger` occurs. Optionally, `min` and/or `max` can be set to limit the subject's value.

`subject` must be an `int` or `float` subject.

If `step` is **negative**, the subject's value will be decremented. Only integer `step` values are supported now.

**Note:** Only integer subjects are supported by `<subject_increment>`.

## 8.2.5 Data binding (Subjects)

### Panoramica

With the help of *Subjects*, the interface of the UI can be created.

A subject is a global data item whose value can be set either from the application or the UI, and whose value can be bound to widget properties.

For example, a `room1_temperature` subject's value can be set in the application when the temperature is measured, and can be bound to a label like this:

```
<lv_label bind_text="room1_temperature"/>
```

### Defining subjects

Subjects can be created in `globals.xml` like this:

```
<globals>
  <subjects>
    <int name="battery_power" value="32"/>
    <string name="user_name" value="John"/>
  </subjects>
</globals>
```

As the example shows, a subject consists of a type, name, and initial value. Currently, only integer and string types are supported.

### Simple binding

Some widgets (e.g., label, slider) support binding the subject's value directly to the widget. These bindings use attributes that start with `bind_*` and reference a subject.

```
<lv_slider bind_value="some_subject"/>
<lv_label bind_text="some_subject"/>
```

Once a binding is created, if the subject's value changes (e.g., by adjusting the slider), all bound widgets will be updated automatically.

### Complex binding

In more complex cases—when a binding requires multiple parameters—the binding can be added as a child element of a widget. This allows binding multiple subjects with different parameters. Ad esempio:

```
<lv_label text="Hello world">
  <lv_obj-bind_flag_if_eq subject="subject1" flag="hidden" ref_value="10"/>
  <lv_obj-bind_flag_if_gt subject="subject1" flag="clickable" ref_value="20"/>
</lv_label>
```

Explanation of complex bindings:

- `<lv_obj-bind_flag_if_eq>`— Set a flag if the subject's value **equals** the reference value.
- `<lv_obj-bind_flag_if_not_eq>`— Set a flag if the subject's value **does not equal** the reference value.

- <lv\_obj-bind\_flag\_if\_gt> — Set a flag if the subject's value is **greater than** the reference value.
- <lv\_obj-bind\_flag\_if\_ge> — Set a flag if the subject's value is **greater than or equal to** the reference value.
- <lv\_obj-bind\_flag\_if\_lt> — Set a flag if the subject's value is **less than** the reference value.
- <lv\_obj-bind\_flag\_if\_le> — Set a flag if the subject's value is **less than or equal to** the reference value.
- <lv\_obj-bind\_state\_if\_eq> — Set a state if the subject's value **equals** the reference value.
- <lv\_obj-bind\_state\_if\_not\_eq> — Set a state if the subject's value **does not equal** the reference value.
- <lv\_obj-bind\_state\_if\_gt> — Set a state if the subject's value is **greater than** the reference value.
- <lv\_obj-bind\_state\_if\_ge> — Set a state if the subject's value is **greater than or equal to** the reference value.
- <lv\_obj-bind\_state\_if\_lt> — Set a state if the subject's value is **less than** the reference value.
- <lv\_obj-bind\_state\_if\_le> — Set a state if the subject's value is **less than or equal to** the reference value.

Note: The `lv_obj`- prefix can be omitted. For example, you can simply write `<bind_state_if_gt>` instead.

## 8.2.6 Translations

### Panoramica

The XML translation module allows defining and using translated strings directly within XML files.

It's built on top of [LVGL's translation module](#). Check it out to learn more about selecting the active language, retrieving translations, and fallback behavior.

### Utilizzo

Example XML translation definition:

```
<translations languages="en de hu">
  <translation tag="dog"      en="The dog"        de="Der Hund"      hu="A kutya"/>
  <translation tag="cat"      en="The cat"        de="Die Katze"    hu="A cicá"/>
  <translation tag="snake"   en="A snake"       de="Eine Schlange" hu="A kígyó"/>
</translations>
```

In the root `<translations>` tag, the `languages` attribute defines the available languages, e.g., `languages="en de hu"`. Language codes are free-form, but ISO-style codes are recommended.

Each `<translation>` defines a `tag`, which acts as the lookup key, and attributes for each language.

Translations may be omitted—[:ref:Fallbacks](#) `<xml_translations_fallback>` will be applied when needed.

To register XML translations:

- `lv_xml_translation_register_from_file("path/to/file.xml")`
- `lv_xml_translation_register_from_data(xml_string)`

Multiple XML sources can be registered; they will be merged and searched collectively.

### Usage in XML

Some widget properties support a `*-translated` suffix to refer to translation tags. Ad esempio:

```
<lv_label text-translated="dog"/>
```

This sets the label's text to the translated string for "dog".

## 8.3 UI Testing

### 8.3.1 Panoramica

The XML test module is a powerful and flexible way to define functional UI tests.

Test XML files are similar to components but are wrapped in a `<test>` tag and consist of two main parts:

- **UI Definition:** Use `<styles>`, `<consts>`, and `<view>` to define how the UI should look. This is identical to how `<component>`s are structured.
- **Test Steps:** Encapsulated in a `<steps>` tag, these define the actions and assertions for the test.

#### Step Types

The following step types are currently supported (with example parameters):

- `<move_to x="10" y="30"/>`: Move the mouse to the specified screen coordinates.
- `<click_at x="10" y="30"/>`: Simulates a click at the specified screen coordinates.
- `<press/>`: Press on the current screen coordinate.
- `<release/>`: Release on the current screen coordinate.
- `<wait ms="100"/>`: Waits for the given number of milliseconds. LVGL continues running, including animations and timers.
- `<freeze ms="100"/>`: Pauses the UI and LVGL's internal time. Useful for visual debugging.
- `<subject_set subject="subject1" value="5">` Set an integer or string subject's value.
- `<subject_compare subject="subject1" value="10">` Compare an integer or strings subject's value against a reference value.
- `<screenshot_compare path="path/to/image.png"/>`: Compares the current screen with a reference image. - If the image doesn't exist, it is created. - If the comparison fails, an image with `_err` suffix is saved for inspection.

Esempio:

```
<test width="300" height="400">
  <view width="100%" height="100%" flex_flow="column">
    <lv_checkbox text="First one!"/>
    <lv_slider />
  </view>

  <steps>
    <screenshot_compare path="imgs/before.png"/>
    <click_at x="32" y="32"/>
    <subject_compare subject="subject2" value="50"/>
    <click_at x="50" y="64"/>
    <subject_set subject="subject2" value="10"/>
    <wait ms="300"/>
    <screenshot_compare path="imgs/after.png"/>
  </steps>
</test>
```

#### Registering Tests

Tests can be registered in two ways:

- From file: `lv_xml_test_register_from_file("path/to/test.xml", "A:ref_images")`
- From string: `lv_xml_test_register_from_data(xml_data, "A:ref_images")`

The second parameter specifies a prefix for screenshot comparison paths.

To unregister tests, use:

- `lv_xml_test_unregister()`

This is called automatically when registering a new test, so manual calls are usually unnecessary.

## Running Tests

### All steps

To execute the registered test, use `lv_xml_test_run_all(slowdown)`.

It will clean the screen and create a fresh instance of the `view` to be tested.

The `slowdown` parameter controls playback speed:

- 0: Maximum speed
- 1: Real-time speed
- 2: Half-speed
- 10: 10× slower
- ...and so on

`lv_xml_test_run_all()` blocks until all steps are completed. It is safe to call it multiple times.

The return value is the number of failed tests.

### Step-by-step

It's also possible to run each step separately, one after another.

First, call `lv_xml_test_run_init()` to prepare for executing the steps. It will clean the screen and create a fresh instance of the `view` to be tested.

Get the number of steps using `lv_xml_test_get_step_count()` and then call `lv_xml_test_run_next(slowdown)` as many times.

`lv_xml_test_run_next()` returns `true` if the given step passed, or `false` if it failed.

Finally, call `lv_xml_test_run_stop()` to clean up and exit testing mode.

## Getting the Test Results

Currently, only `screenshot_compare` steps can fail. However, the result of each step can be queried:

```
uint32_t step_cnt = lv_xml_test_get_step_count();
for(uint32_t i = 0; i < step_cnt; i++) {
    if(lv_xml_test_get_step_type(i) == LV_XML_TEST_STEP_TYPE_SCREENSHOT_COMPARE) {
        if(lv_xml_test_get_status(i))
            printf("Step %d passed\n", i);
        else
            printf("Step %d failed\n", i);
    } else {
        printf("Step %d is not a screenshot comparison\n", i);
    }
}
```

## 8.3.2 API

`lv_test_display.h`  
`lv_test_helpers.h`  
`lv_test_indev.h`  
`lv_test_indev_gesture.h`  
`lv_test_private.h`  
`lv_test_screenshot_compare.h`

## 8.4 Licenza

**Version 1.0 – June 2025**

**Copyright (c) 2025 LVGL LLC**

### 8.4.1 1. Panoramica

The LVGL XML Specification (the "Specification") defines an XML-based language for describing user interfaces within the LVGL ecosystem.

This Specification is protected by copyright. It is provided with limited rights of use as outlined below.

The purpose of this license is to ensure that the Specification remains freely usable for application developers working with LVGL, while restricting its use in third-party UI editors or design tools. This protects the long-term integrity and sustainability of the LVGL ecosystem by preventing unlicensed tools from reusing or replicating the Specification in ways that compete with official LVGL offerings.

### 8.4.2 2. Permitted Usage

You are permitted to:

- Use this Specification in **any firmware or embedded application** built on top of the official **LVGL library** (MIT-licensed).
- Use the LVGL XML loader freely in accordance with its MIT license.
- Use the Specification with the **LVGL UI editor** and **official LVGL tools such as the CLI or XML loader**.
- Write, load, edit, or generate user interfaces described in the LVGL XML Specification for **internal use only**, including automation or configuration tools used solely inside your organization.
- Create **internal scripts or plugins** that help your own developers use or manage this Specification, as long as these are **not shared outside your organization or made publicly available**.
- Share XML-based UI files within customer firmware or projects that target LVGL-based embedded systems.

### 8.4.3 3. Restrictions

You are **not permitted** to:

- Create, release, or distribute any **UI editor, visual builder, layout designer, code generator, or tool** that reads, writes, or interprets XML files using this Specification — **whether commercial, open-source, or intended for public use, use by customers or partners, or any use outside your own organization** — without written permission from LVGL LLC.
- Implement or extend this Specification in any **software or platform that provides UI creation, editing, or design capabilities similar to those in the LVGL UI Editor**.
- Build public APIs, plugins, converters, or SDKs based on this Specification.
- Share or publish internal tooling built around this Specification.
- Use this Specification into general-purpose design tools or frameworks outside the LVGL ecosystem.

These restrictions apply regardless of whether the software is commercial, free, or open-source.

### 8.4.4 4. Clarification for LVGL Users

You are **explicitly permitted** to:

- Use XML files with this Specification in your own projects.
- Load XMLs at runtime using LVGL's XML loader.
- Generate XMLs using the LVGL UI editor.
- Customize XMLs manually or with internal utilities.

- Share LVGL-based projects with XML files as part of a commercial product or firmware.

If you are building embedded software using LVGL, **this license does not affect you**. This license only applies to **UI development tools that read, generate, or process XML files using this Specification for UI development purposes** in a public or commercial manner.

You may create **internal tools** to aid development using the Specification, but these must not be published, sold, or shared externally without a license.

#### **8.4.5 5. Trademarks**

This license does not grant any rights to use the LVGL name, logo, or trademarks.

#### **8.4.6 6. Licensing and Permissions**

To request permission for broader use or commercial licensing (e.g. using this Specification in your own UI tool), contact:

**lvgl@lvgl.io**

All rights not expressly granted herein are reserved by LVGL LLC.

# CAPITOLO 9

## Moduli Ausiliari

### 9.1 Esplora File

`lv_file_explorer` fornisce un'interfaccia utente che consente all'utente finale di esplorare il contenuto di un file system. La sua area principale è chiamata "Browsing Area" [Area di Navigazione] e fornisce l'elenco dei file contenuti nella directory attualmente visualizzata.

Se abilitato, sulla sinistra è presente anche un pannello di "Accesso rapido", che offre un modo comodo per raggiungere le parti del file system a cui si accede di frequente. Available "Quick-Access" destinations are:

- File System,
- HOME,
- Video,
- Pictures,
- Music, and
- Documents.

You specify what paths these lead to during `lv_file_explorer`'s initialization.

`lv_file_explorer` only provides the file browsing and events caused by user activity (e.g. clicking a file), but does not provide the actual file operations. Client code must hook various events and decide what to do when they are emitted (e.g. a click or double-click on a file). The actions taken might to open the file, display it, send it to some other part of the application, etc.. `lv_file_explorer` passes the full path and name of file that was clicked to the event callback functions. What happens next is up to the application designer.

`lv_file_explorer` uses the *Table* (`lv_table`) Widget for the "Browsing Area", and the *List* (`lv_list`) Widget for the "Quick-Access" panel when it is enabled. Thus, `LV_USE_TABLE` macro must be set to a non-zero value in `lv.conf.h` in order to use `lv_file_explorer`, and `LV_USE_LIST` must be set to a non-zero value to use the "Quick-Access" panel.

#### Nota

In order to use File Explorer, *File System* (`lv_fs_drv`) has to be set up and know about all the drive letters you use when passing paths to File System (described below).

### 9.1.1 Prerequisiti

If you haven't already done so, you will need to learn about the LVGL *File System abstraction*, since it must be set up and be functional for File Explorer to work.

### 9.1.2 Utilizzo

Set `LV_USE_FILE_EXPLORER` to a non-zero value in `lv_conf.h`.

First use `lv_file_explorer_create(lv_screen_active())` to create a File Explorer. The default size is the screen size. After that, you can customize the style like any Widget.

The size of the `current_path` buffer is set by `LV_FILE_EXPLORER_PATH_MAX_LEN` in `lv_conf.h`.

The object hierarchy of a freshly-created File Explorer looks like this:

- **File Explorer**: occupies full area of parent Widget, typically a Screen (Flex-Flow COLUMN)
  - **Container**: occupies full area of File Explorer (Flex grow 1)
    - \* **Quick-Access Panel**:
      - **Device List**: grows to accommodate children
      - **File System**: button
      - **Places List**: grows to accommodate children
      - **HOME**: button
      - **Video**: button
      - **Pictures**: button
      - **Music**: button
      - **Documents**: button
    - \* **Browser Panel**:
      - **Header**: 14% of Browser Panel height
      - **Current Path**: label
      - **File Table**: with 1 column, 86% of Browser Panel height
  - **Campi**:
    - \* `home_dir=NULL`
    - \* `video_dir=NULL`
    - \* `pictures_dir=NULL`
    - \* `music_dir=NULL`
    - \* `docs_dir=NULL`
    - \* `fs_dir=NULL`
    - \* `current_path=[empty buffer]`
    - \* `sel_fn` (selected file)
    - \* `sort` (default `LV_EXPLORER_SORT_NONE`)

### Accessing the Parts

This list of functions provides access to the parts shown in diagram above:

- `lv_file_explorer_get_selected_file_name(explorer)` (pointer to NUL-terminated string containing file-path user selected; typically used inside an `LV_EVENT_CLICKED` event)

- `lv_file_explorer_get_current_path(explorer)` (pointer to `current_path` char buffer)
- `lv_file_explorer_get_file_table(explorer)` (pointer to `File Table Table (lv_table)` Widget)
- `lv_file_explorer_get_header(explorer)` (pointer to Header `Widget Base (lv_obj)` Widget)
- `lv_file_explorer_get_path_label(explorer)` (pointer to Current Path Label `Label (lv_label)` Widget)
- `lv_file_explorer_get_quick_access_area(explorer)` (pointer to Quick-Access Panel `Widget Base (lv_obj)`)
- `lv_file_explorer_get_places_list(explorer)` (pointer to Places List `List (lv_list)` Widget)
- `lv_file_explorer_get_device_list(explorer)` (pointer to Device List `List (lv_list)` Widget)

### Quick-Access Panel

The Quick-Access Panel behaves like a typical navigation panel and appears on the left, while the Browser Panel appears on the right

This panel is optional. If you set `LV_FILE_EXPLORER_QUICK_ACCESS` to 0 in `lv.conf.h`, the Quick-Access Panel will not be created. This saves only a little bit of memory.

Soon after the File Explorer is created, you typically use `lv_file_explorer_set_quick_access_path(explorer, LV_EXPLORER_XXX_DIR, "path")` to set the path that will be navigated to when the buttons in the Quick-Access Panel are clicked, which is currently a fixed list. The corresponding values you will need to pass as the 2nd argument are the following:

- `LV_EXPLORER_HOME_DIR`
- `LV_EXPLORER_MUSIC_DIR`
- `LV_EXPLORER_PICTURES_DIR`
- `LV_EXPLORER_VIDEO_DIR`
- `LV_EXPLORER_DOCS_DIR`
- `LV_EXPLORER_FS_DIR`

### Sort

You can use `lv_file_explorer_set_sort(explorer, LV_EXPLORER_SORT_XX)` to set the sorting method.

These are the possible sorting methods:

- `LV_EXPLORER_SORT_NONE` (default)
- `LV_EXPLORER_SORT_KIND`

`lv_file_explorer_get_sort(explorer)` returns the current sorting method.

### 9.1.3 Eventi

- `LV_EVENT_READY` Sent when a directory is opened, which can happen:
  - when the File Explorer is initially opened,
  - after a user clicks on a Quick-Access Panel navigation button, and
  - after the user clicks on a directory displayed in the Browser Panel.

You can use it to, for example, customize the file sort.

- ***LV\_EVENT\_VALUE\_CHANGED*** Sent once when any item (file) in the **Browser Panel**'s file list is clicked.
- ***LV\_EVENT\_CLICKED*** Sent twice when an item in the **Browser Panel** is clicked: once as a result of the input-device ***LV\_EVENT\_RELEASED*** event and a second as a result of the input device ***LV\_EVENT\_CLICKED*** event. This applies to files, directories, and the "< Back" item in the **Browser Panel**.

In these events you can use `lv_file_explorer_get_current_path()` to get the current path and `lv_file_explorer_get_selected_file_name()` to get the name of the currently selected file in the event processing function. Ad esempio:

```
static void file_explorer_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        char * cur_path = lv_file_explorer_get_current_path(widget);
        char * sel_fn = lv_file_explorer_get_selected_file_name(widget);
        LV_LOG_USER("%s%s", cur_path, sel_fn);
    }
}
```

You can also save the obtained **path** and **file** name into an array through functions such as `strcpy()` and `strcat()` for later use.

## 9.1.4 Esempio

### Simple File Explorer

```
#include "../../lv_examples.h"

#if LV_USE_TABLE && LV_USE_FILE_EXPLORER && (LV_USE_FS_STDIO || LV_USE_FS_POSIX || LV_USE_FS_WIN32 || LV_USE_FS_FATFS) && LV_BUILD_EXAMPLES
#include <stdlib.h>
#include <string.h>

static void file_explorer_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        const char * cur_path = lv_file_explorer_get_current_path(obj);
        const char * sel_fn = lv_file_explorer_get_selected_file_name(obj);
        LV_LOG_USER("%s%s", cur_path, sel_fn);
    }
}

void lv_example_file_explorer_1(void)
{
    lv_obj_t * file_explorer = lv_file_explorer_create(lv_screen_active());
    lv_file_explorer_set_sort(file_explorer, LV_EXPLORER_SORT_KIND);

#if LV_USE_FS_WIN32
/* Note to Windows users: the initial "C:" on these paths corresponds to
 * the value of 'LV_FS_WIN32_LETTER' in 'lv_conf.h', and should not be
 * confused with the Windows/DOS drive letter. It is an identifier that
 * is used to enable LVGL to look up the appropriate driver from a list of
 * registered file-system drivers. 'lv_fs_win32_init()' happens to use the
 * identifier letter 'C' so "C:" is the driver-identifier-prefix used here.
 * The "C:" following that is indeed the Windows/DOS drive letter and is
 * part of the actual path that gets passed to the OS-level functions.
 *
 * See https://docs.lvgl.io/master/details/main-components/fs.html for details.
 * File Explorer uses `lv_fs` internally, thus the required prefix in path strings.
 */
    lv_file_explorer_open_dir(file_explorer, "C:C:/");
#endif

#if LV_FILE_EXPLORER_QUICK_ACCESS
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, "C:C:/Users/Public/Desktop");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, "C:C:/Users/Public/Videos");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, "C:C:/Users/Public/Pictures");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, "C:C:/Users/Public/Music");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, "C:C:/Users/Public/Documents");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "C:C:/");
#endif

#else
/* linux */
    lv_file_explorer_open_dir(file_explorer, "A:/");

#if LV_FILE_EXPLORER_QUICK_ACCESS
    const char * envvar = "HOME";
    char home_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(home_dir, "A:");
    /* get the user's home directory from the HOME environment variable*/
    strcat(home_dir, getenv(envvar));
    LV_LOG_USER("home_dir: %s\n", home_dir);
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, home_dir);
    char video_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(video_dir, home_dir);
    strcat(video_dir, "/Videos");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, video_dir);
    char picture_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(picture_dir, home_dir);
    strcat(picture_dir, "/Pictures");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, picture_dir);

```

(continues on next page)

(continua dalla pagina precedente)

```

char music_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(music_dir, home_dir);
strcat(music_dir, "/Music");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, music_dir);
char document_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(document_dir, home_dir);
strcat(document_dir, "/Documents");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, document_dir);

lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "A:/");
#endif
#endif

lv_obj_add_event_cb(file_explorer, file_explorer_event_handler, LV_EVENT_ALL, NULL);
}
#endif

```

## Control File Explorer

```

#include "../../../lv_examples.h"

#if LV_USE_TABLE && LV_USE_DROPDOWN && LV_USE_FILE_EXPLORER && (LV_USE_FS_STDIO || LV_USE_FS_POSIX || LV_USE_FS_WIN32 || LV_USE_FS_FATFS) && LV_BUILD_
→EXAMPLES

#include <stdlib.h>
#include <string.h>

static void file_explorer_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        const char * cur_path = lv_file_explorer_get_current_path(obj);
        const char * sel_fn = lv_file_explorer_get_selected_file_name(obj);

        LV_LOG_USER("%s%s", cur_path, sel_fn);
    }
}

#if LV_FILE_EXPLORER_QUICK_ACCESS
static void btn_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * btn = lv_event_get_target_obj(e);
    lv_obj_t * file_explorer = (lv_obj_t *) lv_event_get_user_data(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        if(lv_obj_has_state(btn, LV_STATE_CHECKED))
            lv_obj_add_flag(file_explorer, LV_OBJ_FLAG_HIDDEN);
        else
            lv_obj_remove_flag(file_explorer, LV_OBJ_FLAG_HIDDEN);
    }
}

static void dd_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * dd = lv_event_get_target_obj(e);
    lv_obj_t * fe_quick_access_obj = (lv_obj_t *) lv_event_get_user_data(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
        lv_dropdown_get_selected_str(dd, buf, sizeof(buf));
        if(strcmp(buf, "NONE") == 0) {
            lv_file_explorer_set_sort(fe_quick_access_obj, LV_EXPLORER_SORT_NONE);
        } else if(strcmp(buf, "KIND") == 0) {
            lv_file_explorer_set_sort(fe_quick_access_obj, LV_EXPLORER_SORT_KIND);
        }
    }
}
#endif

void lv_example_file_explorer_2(void)
{
    lv_obj_t * file_explorer = lv_file_explorer_create(lv_screen_active());

#if LV_USE_FS_WIN32
/* Note to Windows users: the initial "C:" on these paths corresponds to
 * the value of `LV_FS_WIN32 LETTER` in `lv_conf.h`, and should not be
 * confused with the Windows/DOS drive letter. It is an identifier that
 * is used to enable LVGL to look up the appropriate driver from a list of
 * registered file-system drivers. `lv_fs_win32_init()` happens to use the
 * identifier letter 'C' so "C:" is the driver-identifier-prefix used here.
 * The "C:" following that is indeed the Windows/DOS drive letter and is
 * part of the actual path that gets passed to the OS-level functions.
 *
 * See https://docs.lvgl.io/master/details/main-components/fs.html for details.
 * File Explorer uses 'lv_fs' internally, thus the required prefix in path strings.
 */
lv_file_explorer_open_dir(file_explorer, "C:C:/");
#endif

#if LV_FILE_EXPLORER_QUICK_ACCESS
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, "C:C:/Users/Public/Desktop");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, "C:C:/Users/Public/Videos");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, "C:C:/Users/Public/Pictures");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, "C:C:/Users/Public/Music");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, "C:C:/Users/Public/Documents");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "C:C:/");
#endif

#else
/* linux */
lv_file_explorer_open_dir(file_explorer, "A:/");
#endif
#if LV_FILE_EXPLORER_QUICK_ACCESS
const char * envvar = "HOME";

```

(continues on next page)

(continua dalla pagina precedente)

```

char home_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(home_dir, "A:");
/* get the user's home directory from the HOME environment variable/
strcat(home_dir, getenv(envvar));
LV_LOG_USER("home_dir: %s\n", home_dir);
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, home_dir);
char video_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(video_dir, home_dir);
strcat(video_dir, "/Videos");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, video_dir);
char picture_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(picture_dir, home_dir);
strcat(picture_dir, "/Pictures");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, picture_dir);
char music_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(music_dir, home_dir);
strcat(music_dir, "/Music");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, music_dir);
char document_dir[LV_FS_MAX_PATH_LENGTH];
strcpy(document_dir, home_dir);
strcat(document_dir, "/Documents");
lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, document_dir);

lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "A:/");
#endif
#endif

lv_obj_add_event_cb(file_explorer, file_explorer_event_handler, LV_EVENT_ALL, NULL);

#if LV_FILE_EXPLORER_QUICK_ACCESS
/*Quick access status control button*/
lv_obj_t * fe_quick_access_obj = lv_file_explorer_get_quick_access_area(file_explorer);
lv_obj_t * fe_header_obj = lv_file_explorer_get_header(file_explorer);
lv_obj_t * btn = lv_button_create(fe_header_obj);
lv_obj_set_style_radius(btn, 2, 0);
lv_obj_set_style_pad_all(btn, 4, 0);
lv_obj_align(btn, LV_ALIGN_LEFT_MID, 0, 0);
lv_obj_add_flag(btn, LV_OBJ_FLAG_CHECKABLE);

lv_obj_t * label = lv_label_create(btn);
lv_label_set_text(label, LV_SYMBOL_LIST);
lv_obj_center(label);

lv_obj_add_event_cb(btn, btn_event_handler, LV_EVENT_VALUE_CHANGED, fe_quick_access_obj);

/*Sort control*/
static const char * opts = "NONE\n"
                           "KIND";

lv_obj_t * dd = lv_dropdown_create(fe_header_obj);
lv_obj_set_style_radius(dd, 4, 0);
lv_obj_set_style_pad_all(dd, 0, 0);
lv_obj_set_size(dd, LV_PCT(30), LV_SIZE_CONTENT);
lv_dropdown_set_options_static(dd, opts);
lv_obj_align(dd, LV_ALIGN_RIGHT_MID, 0, 0);

lv_obj_add_event_cb(dd, dd_event_handler, LV_EVENT_VALUE_CHANGED, file_explorer);
#endif
}
#endif

```

## Custom sort

```

#include "../../lv_examples.h"

#if LV_USE_TABLE && LV_USE_FILE_EXPLORER && (LV_USE_FS_STDIO || LV_USE_FS_POSIX || LV_USE_FS_WIN32 || LV_USE_FS_FATFS) && LV_BUILD_EXAMPLES

#include <stdlib.h>
#include <string.h>

static void exch_table_item(lv_obj_t * tb, int16_t i, int16_t j)
{
    const char * tmp;
    tmp = lv_table_get_cell_value(tb, i, 0);
    lv_table_set_cell_value(tb, 0, 2, tmp);
    lv_table_set_cell_value(tb, i, 0, lv_table_get_cell_value(tb, j, 0));
    lv_table_set_cell_value(tb, j, 0, lv_table_get_cell_value(tb, 0, 2));

    tmp = lv_table_get_cell_value(tb, i, 1);
    lv_table_set_cell_value(tb, 0, 2, tmp);
    lv_table_set_cell_value(tb, i, 1, lv_table_get_cell_value(tb, j, 1));
    lv_table_set_cell_value(tb, j, 1, lv_table_get_cell_value(tb, 0, 2));
}

/*Quick sort 3 way*/
static void sort_by_file_kind(lv_obj_t * tb, int16_t lo, int16_t hi)
{
    if(lo >= hi) return;

    int16_t lt = lo;
    int16_t i = lo + 1;
    int16_t gt = hi;
    const char * v = lv_table_get_cell_value(tb, lo, 1);
    while(i <= gt) {
        if(strcmp(lv_table_get_cell_value(tb, i, 1), v) < 0)
            exch_table_item(tb, lt++, i++);
        else if(strcmp(lv_table_get_cell_value(tb, i, 1), v) > 0)
            exch_table_item(tb, i, gt--);
        else
            i++;
    }

    sort_by_file_kind(tb, lo, lt - 1);
    sort_by_file_kind(tb, gt + 1, hi);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

static void file_explorer_event_handler(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);

    if(code == LV_EVENT_VALUE_CHANGED) {
        const char * cur_path = lv_file_explorer_get_current_path(obj);
        const char * sel_fn = lv_file_explorer_get_selected_file_name(obj);

        LV_LOG_USER("%s%s", cur_path, sel_fn);
    }
    else if(code == LV_EVENT_READY) {
        lv_obj_t * tb = lv_file_explorer_get_file_table(obj);
        uint16_t sum = lv_table_get_row_count(tb);

        sort_by_file_kind(tb, 0, (sum - 1));
    }
}

void lv_example_file_explorer_3(void)
{
    lv_obj_t * file_explorer = lv_file_explorer_create(lv_screen_active());
    /*Before custom sort, please set the default sorting to NONE. The default is NONE.*/
    lv_file_explorer_set_sort(file_explorer, LV_EXPLORER_SORT_NONE);

#if LV_USE_FS_WIN32
    /* Note to Windows users: the initial "C:" on these paths corresponds to
     * the value of `LV_FS_WIN32_LETTER` in `lv_conf.h`, and should not be
     * confused with the Windows/DOS drive letter. It is an identifier that
     * is used to enable LVGL to look up the appropriate driver from a list of
     * registered file-system drivers. `lv_fs_win32_init()` happens to use the
     * identifier letter 'C' so "C:" is the driver-identifier-prefix used here.
     * The "C:" following that is indeed the Windows/DOS drive letter and is
     * part of the actual path that gets passed to the OS-level functions.
     *
     * See https://docs.lvgl.io/master/details/main-components/fs.html for details.
     * File Explorer uses `lv_fs` internally, thus the required prefix in path strings.
     */
    lv_file_explorer_open_dir(file_explorer, "C:C:/");
#endif

#if LV_FILE_EXPLORER_QUICK_ACCESS
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, "C:C:/Users/Public/Desktop");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, "C:C:/Users/Public/Videos");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, "C:C:/Users/Public/Pictures");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, "C:C:/Users/Public/Music");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, "C:C:/Users/Public/Documents");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "C:C:/");
#endif

#else
    /* linux */
    lv_file_explorer_open_dir(file_explorer, "A:/");
#endif

#if LV_FILE_EXPLORER_QUICK_ACCESS
    const char * envvar = "HOME";
    char home_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(home_dir, "A:/");
    /* get the user's home directory from the HOME environment variable*/
    strcat(home_dir, getenv(envvar));
    LV_LOG_USER("home_dir: %s\n", home_dir);
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_HOME_DIR, home_dir);
    char video_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(video_dir, home_dir);
    strcat(video_dir, "/Videos");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_VIDEO_DIR, video_dir);
    char picture_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(picture_dir, home_dir);
    strcat(picture_dir, "/Pictures");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_PICTURE_DIR, picture_dir);
    char music_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(music_dir, home_dir);
    strcat(music_dir, "/Music");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_MUSIC_DIR, music_dir);
    char document_dir[LV_FS_MAX_PATH_LENGTH];
    strcpy(document_dir, home_dir);
    strcat(document_dir, "/Documents");
    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_DOCS_DIR, document_dir);

    lv_file_explorer_set_quick_access_path(file_explorer, LV_EXPLORER_FS_DIR, "A:/");
#endif
#endif

    lv_obj_add_event_cb(file_explorer, file_explorer_event_handler, LV_EVENT_ALL, NULL);
}

```

## 9.1.5 API

*lv\_file\_explorer.h*

*lv\_file\_explorer\_private.h*

## 9.2 Font Manager

The font manager is a secondary encapsulation of the LVGL font engine, which facilitates the use and management of fonts for applications. The font management functions currently provided include:

- Font resource reference counting (reduces repeated creation of font resources).
- Font resource concatenation (font fallback).

- Font resource recycling mechanism (buffers recently deleted font resources to reduce the time overhead of repeated creation).

### 9.2.1 Utilizzo

Enable FreeType and Font Manager in `lv_conf.h` by setting the `LV_USE_FONT_MANAGER` macros to non-zero values, and configure `LV_FONT_MANAGER_NAME_MAX_LEN` to set the maximum length of the font name.

#### Initialize Font Manager

Use `lv_font_manager_create()` to create a font manager, where the `recycle_cache_size()` parameter is used to set the number of font recycling caches, which can improve font creation efficiency.

Use `lv_font_manager_add_src_static()` to add a mapping between font names and font resources to tell the font manager how to access the font resources. Note that if the font resource description structure is not statically allocated (for example, allocated from a local variable), use `lv_font_manager_add_src()` to add the resource. This function will copy the contents of the structure itself.

Use `lv_font_manager_remove_src()` to remove the font resource mapping.

It should be noted that the `src` parameter must strictly correspond to `class_p`. `class_p` will affect the way the font manager interprets `src`. If an incompatible parameter is passed, the program may fail. For currently supported font classes, please refer to the example code.

```
static lv_font_manager_t * g_font_manager = NULL;

void font_manager_init(void)
{
    /* Create font manager, with 8 fonts recycling buffers */
    g_font_manager = lv_font_manager_create(8);

    /* Add font path mapping to font manager */
    lv_font_manager_add_src_static(g_font_manager, "Lato-Regular", "./lvgl/examples/libs/freetype/Lato-Regular.ttf", &lv_freetype_font_class);

    char path[] = "/path/to/myfont.ttf";
    lv_font_manager_add_src(g_font_manager, "MyFont", path, &lv_freetype_font_class);
}
```

#### Create Font from Font Manager

The parameters will be passed to the font creation function of the font backend, such as `lv_freetype_font_create()` and `lv_tiny_ttf_create_file()`. The font backend will select the supported parameters by itself and ignore the unsupported parameters.

The `font_family` parameter can be filled with the names of multiple fonts (separated by `,`) to achieve font concatenation (when the corresponding glyph is not found in a font file, it will automatically search from the next concatenated font).

```
static lv_font_t * g_font = NULL;

/* Create font from font manager */
lv_font_t * g_font = lv_font_manager_create_font(g_font_manager,
                                                "Lato-Regular,MyFont",
                                                LV_FREETYPE_FONT_RENDER_MODE_BITMAP,
                                                24,
                                                LV_FREETYPE_FONT_STYLE_NORMAL,
                                                LV_FONT_KERNING_NONE);

/* Handle error case */
if(g_font == NULL) {
    g_font = (lv_font_t *)LV_FONT_DEFAULT;
}

/* Create label with the font */
lv_obj_t * label = lv_label_create(lv_screen_active());
lv_obj_set_style_text_font(label, g_font, 0);
lv_label_set_text(label, "Hello World!");
```

#### Delete Font

Use `lv_font_manager_delete_font()` to delete the font when it is no longer needed. The font manager will mark the font resource as a recyclable font so that it has the possibility of being more quickly created next time.

Note that you need to delete any Widgets that used the font first, and then delete the font to avoid accessing wild pointers.

```
/* Delete label and font */
lv_obj_delete(label);
lv_font_manager_delete_font(g_font_manager, g_font);
g_font = NULL;
```

## Delete Font Manager

Use `lv_font_manager_delete()` to destroy the entire font manager. It should be noted that before destruction, it is necessary to ensure that the application has deleted all fonts using `lv_font_manager_delete_font()`. The font manager will check the reference status of all allocated fonts. If there are still fonts being referenced, the font manager will fail to be destroyed and the function will return false.

### 9.2.2 Esempio

#### Create FreeType Font

```
#include "../../lv_examples.h"
#if LV_USE_FONT_MANAGER && LV_USE_FREETYPE && LV_BUILD_EXAMPLES

#if LV_FREETYPE_USE_LVGL_PORT
#define PATH_PREFIX "A:"
#else
#define PATH_PREFIX "./"
#endif

static lv_font_manager_t * g_font_manager = NULL;

void lv_example_font_manager_1(void)
{
    /* Create font manager, with 8 fonts recycling buffers */
    g_font_manager = lv_font_manager_create(8);

    /* Add font path mapping to font manager */
    lv_font_manager_add_src_static(g_font_manager,
        "Lato-Regular",
        PATH_PREFIX "lvgl/examples/libs/freetype/Lato-Regular.ttf",
        &lv_freetype_font_class);

    /* Create font from font manager */
    lv_font_t * font = lv_font_manager_create_font(g_font_manager,
        "Lato-Regular",
        LV_FREETYPE_FONT_RENDER_MODE_BITMAP,
        24,
        LV_FREETYPE_FONT_STYLE_NORMAL,
        LV_FONT_KERNING_NONE);

    if(!font) {
        LV_LOG_ERROR("Could not create font");
        return;
    }

    /* Create label with the font */
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_style_text_font(label, font, 0);
    lv_label_set_text(label, "Hello Font Manager!");
    lv_obj_center(label);
}

#else

void lv_example_font_manager_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Font Manager or FreeType is not enabled");
    lv_obj_center(label);
}

#endif
```

#### Create Font Family

```
#include "../../lv_examples.h"
#if LV_USE_FONT_MANAGER && LV_BUILD_EXAMPLES

#if (LV_FREETYPE_USE_LVGL_PORT) && LV_FREETYPE_USE_LVGL_PORT
#define PATH_PREFIX "A:"
#else
#define PATH_PREFIX "./"
#endif

static lv_font_manager_t * g_font_manager = NULL;

void lv_example_font_manager_2(void)
{
    /* Create font manager, with 8 fonts recycling buffers */
    g_font_manager = lv_font_manager_create(8);

#if LV_FONT_MONTserrat_14 && LV_FONT_MONTserrat_32
    /* Register built-in font sources */
    static lv_builtin_font_src_t builtin_font_src[3] = { 0 };
    builtin_font_src[0].font_p = &lv_font_montserrat_14;
    builtin_font_src[0].size = 14;
    builtin_font_src[1].font_p = &lv_font_montserrat_32;
    builtin_font_src[1].size = 32;

    /* IMPORTANT! Marking the end of the array */

```

(continues on next page)

(continua dalla pagina precedente)

```

builtin_font_src[2].font_p = NULL;
builtin_font_src[2].size = 0;

lv_font_manager_add_src_static(g_font_manager,
                               "Montserrat",
                               builtin_font_src,
                               &lv_builtin_font_class);

#endif

#if LV_USE_FREETYPE
/* Register FreeType font source */
lv_font_manager_add_src_static(g_font_manager,
                               "NotoColorEmoji",
                               PATH_PREFIX "lvg/ examples/libs/freetype/NotoColorEmoji-32.subset.ttf",
                               &lv_freetype_font_class);
#endif

#if LV_USE_TINY_TTF && LV_TINY_TTF_FILE_SUPPORT
/* Register TinyTTF font source */
static const lv_tiny_ttf_font_src_t tiny_ttf_font_src = {
    .path = "A:lvg/ examples/libs/tiny_ttf/Ubuntu-Medium.ttf",
    .data = NULL,
    .data_size = 0,
    .cache_size = 0,
};

lv_font_manager_add_src_static(g_font_manager,
                               "Ubuntu-Medium",
                               &tiny_ttf_font_src,
                               &lv_tiny_ttf_font_class);

#endif

/* Create font from font manager */
lv_font_t * font = lv_font_manager_create_font(g_font_manager,
                                              "Ubuntu-Medium.NotoColorEmoji.Montserrat",
                                              LV_FREETYPE_FONT_RENDER_MODE_BITMAP,
                                              32,
                                              LV_FREETYPE_FONT_STYLE_NORMAL,
                                              LV_FONT_KERNING_NONE);

if(!font) {
    LV_LOG_ERROR("Could not create font");
    return;
}

/* Create label with the font */
lv_obj_t * label = lv_label_create(lv_screen_active());
lv_obj_set_style_text_font(label, font, 0);
lv_label_set_text(label, "Hello Font Manager! ☺ " LV_SYMBOL_OK);
lv_obj_center(label);
}

#else

void lv_example_font_manager_2(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Font Manager is not enabled");
    lv_obj_center(label);
}

#endif

```

## Create Custom Image Font

```

#include "../../lv_examples.h"
#if LV_USE_FONT_MANAGER && LV_USE_FREETYPE && LV_USE_IMGFONT && LV_BUILD_EXAMPLES

#if LV_FREETYPE_USE_LVGL_PORT
#define PATH_PREFIX "A:"
#else
#define PATH_PREFIX "./"
#endif

typedef struct {
    uint16_t match_size_min;
    uint16_t match_size_max;
    lv_imgfont_get_path_cb_t get_path_cb;
} imgfont_src_t;

static lv_font_manager_t * g_font_manager = NULL;

static const void * get_imgfont_path(const lv_font_t * font, uint32_t unicode, uint32_t unicode_next,
                                     int32_t * offset_y, void * user_data)
{
    LV_UNUSED(font);
    LV_UNUSED(unicode_next);
    LV_UNUSED(offset_y);
    LV_UNUSED(user_data);

    LV_IMAGE_DECLARE(emoji_F617);

    if(unicode == 0xF617) {
        return &emoji_F617;
    }

    return NULL;
}

static lv_font_t * imgfont_create_cb(const lv_font_info_t * info, const void * src)
{
    const imgfont_src_t * imgfont_src = (imgfont_src_t *) src;

    if(info->size < imgfont_src->match_size_min
       || info->size > imgfont_src->match_size_max) {
        return NULL;
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

    return lv_imgfont_create(info->size, imgfont_src->get_path_cb, NULL);
}

static void imgfont_delete_cb(lv_font_t * font)
{
    lv_imgfont_destroy(font);
}

static void * imgfont_dup_src_cb(const void * src)
{
    imgfont_src_t * imgfont_src = (imgfont_src_t *) lv_malloc(sizeof(imgfont_src_t));
    LV_ASSERT_MALLOC(imgfont_src);
    lv_memcpy(imgfont_src, src, sizeof(imgfont_src_t));
    return imgfont_src;
}

static void imgfont_free_src_cb(void * src)
{
    lv_free(src);
}

static const lv_font_class_t imgfont_class = {
    .create_cb = imgfont_create_cb,
    .delete_cb = imgfont_delete_cb,
    .dup_src_cb = imgfont_dup_src_cb,
    .free_src_cb = imgfont_free_src_cb,
};

void lv_example_font_manager_3(void)
{
    /* Create font manager, with 8 fonts recycling buffers */
    g_font_manager = lv_font_manager_create(8);

    /* Add font path mapping to font manager */
    lv_font_manager_add_src_static(g_font_manager,
        "Lato-Regular",
        PATH_PREFIX "lvgl/examples/libs/freetype/Lato-Regular.ttf",
        &lv_freetype_font_class);

    /* Add custom imgfont source to font manager */
    static const imgfont_src_t imgfont_src = {
        .match_size_min = 70,
        .match_size_max = 80,
        .get_path_cb = get_imgfont_path,
    };
    lv_font_manager_add_src_static(g_font_manager,
        "Emoji",
        &imgfont_src,
        &imgfont_class);

    /* Create font from font manager */
    lv_font_t * font = lv_font_manager_create_font(g_font_manager,
        "Lato-Regular,Emoji",
        LV_FREETYPE_FONT_RENDER_MODE_BITMAP,
        75,
        LV_FREETYPE_FONT_STYLE_NORMAL,
        LV_FONT_KERNING_NONE);

    if(!font) {
        LV_LOG_ERROR("Could not create font");
        return;
    }

    /* Create label with the font */
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_style_text_font(label, font, 0);
    lv_label_set_text(label, "Quiet\u00d7");
    lv_obj_center(label);
}

#else

void lv_example_font_manager_3(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Font Manager or Freetype or Imgfont is not enabled");
    lv_obj_center(label);
}

#endif

```

### 9.2.3 API

*lv\_font\_manager.h*

*lv\_font\_manager\_recycle.h*

## 9.3 Fragment

Fragment is a concept copied from [Android](#).

It represents a reusable portion of your app's UI. A fragment defines and manages its own layout, has its own lifecycle, and can handle its own events. Like Android's Fragment that must be hosted by an activity or another fragment, Fragment in LVGL needs to be hosted by a Widget, or another fragment. The fragment's view hierarchy becomes part of, or attaches to, the host's view hierarchy.

Such concept also has some similarities to [UIViewController](#) on iOS.

Fragment Manager is a manager holding references to fragments attached to it, and has an internal stack to achieve forward and backwards navigation. You can use fragment manager to build a navigation stack, or a multi-pane application easily.

### 9.3.1 Utilizzo

Enable `LV_USE_FRAGMENT` in `lv_conf.h`.

#### Create Fragment Class

```
struct sample_fragment_t {
    /* IMPORTANT: don't miss this part */
    lv_fragment_t base;
    /* States, object references and data fields for this fragment */
    const char *title;
};

const lv_fragment_class_t sample_cls = {
    /* Initialize something needed */
    .constructor_cb = sample_fragment_ctor,
    /* Create view objects */
    .create_obj_cb = sample_fragment_create_obj,
    /* IMPORTANT: size of your fragment struct */
    .instance_size = sizeof(struct sample_fragment_t),
};
```

#### Use `lv_fragment_manager`

```
/* Create fragment instance, and Widgets will be added to container */
lv_fragment_manager_t *manager = lv_fragment_manager_create(container, NULL);
/* Replace current fragment with instance of sample_cls, and init_argument is user defined pointer */
lv_fragment_manager_replace(manager, &sample_cls, init_argument);
```

#### Fragment Based Navigation

```
/* Add one instance into manager stack. View object of current fragment will be destroyed,
 * but instances created in class constructor will be kept.
 */
lv_fragment_manager_push(manager, &sample_cls, NULL);

/* Remove the top most fragment from the stack, and bring back previous one. */
lv_fragment_manager_pop(manager);
```

### 9.3.2 Esempio

#### Basic fragment usage

```
/** @file lv_example_fragment_1.c
 * @brief Basic usage of obj fragment
 */
#include "../lv_examples.h"

#if LV_USE_FRAGMENT && LV_BUILD_EXAMPLES

static void sample_fragment_ctor(lv_fragment_t * self, void * args);

static lv_obj_t * sample_fragment_create_obj(lv_fragment_t * self, lv_obj_t * parent);

static void sample_container_delete(lv_event_t * e);

static lv_obj_t * root = NULL;

struct sample_fragment_t {
    lv_fragment_t base;
    const char * name;
};

static const lv_fragment_class_t sample_cls = {
    .constructor_cb = sample_fragment_ctor,
    .create_obj_cb = sample_fragment_create_obj,
    .instance_size = sizeof(struct sample_fragment_t),
};

void lv_example_fragment_1(void)
{
    root = lv_obj_create(lv_screen_active());
    lv_obj_set_size(root, LV_PCT(100), LV_PCT(100));
    lv_fragment_manager_t * manager = lv_fragment_manager_create(NULL);
    /* Clean up the fragment manager before objects in containers got deleted */
    lv_obj_add_event_cb(root, sample_container_delete, LV_EVENT_DELETE, manager);

    lv_fragment_t * fragment = lv_fragment_create(&sample_cls, (void *) "Fragment");
    lv_fragment_manager_replace(manager, fragment, &root);
}

static void sample_fragment_ctor(lv_fragment_t * self, void * args)
{
    ((struct sample_fragment_t *) self)->name = (const char *) args;
}
```

(continues on next page)

(continua dalla pagina precedente)

```

static lv_obj_t * sample_fragment_create_obj(lv_fragment_t * self, lv_obj_t * parent)
{
    lv_obj_t * label = lv_label_create(parent);
    lv_obj_set_style_bg_opa(label, LV_OPA_COVER, 0);
    lv_label_set_text_fmt(label, "Hello, %s!", ((struct sample_fragment_t *) self)->name);
    return label;
}

static void sample_container_delete(lv_event_t * e)
{
    lv_fragment_manager_t * manager = (lv_fragment_manager_t *) lv_event_get_user_data(e);
    lv_fragment_manager_delete(manager);
}

#endif

```

## Stack navigation example

```

/** @file lv_example_fragment_2.c
 * @brief Navigation stack using obj fragment
 */
#include "../../../lv_examples.h"

#if LV_USE_FRAGMENT && LV_USE_WIN && LV_USE_GRID && LV_BUILD_EXAMPLES

static void sample_fragment_ctor(lv_fragment_t * self, void * args);

static lv_obj_t * sample_fragment_create_obj(lv_fragment_t * self, lv_obj_t * parent);

static void sample_push_click(lv_event_t * e);

static void sample_pop_click(lv_event_t * e);

static void sample_container_delete(lv_event_t * e);

static void sample_fragment_inc_click(lv_event_t * e);

typedef struct sample_fragment_t {
    lv_fragment_t base;
    lv_obj_t * label;
    int depth;
    int counter;
} sample_fragment_t;

static const lv_fragment_class_t sample_cls = {
    .constructor_cb = sample_fragment_ctor,
    .create_obj_cb = sample_fragment_create_obj,
    .instance_size = sizeof(sample_fragment_t),
};

static lv_obj_t * container = NULL;

void lv_example_fragment_2(void)
{
    lv_obj_t * root = lv_obj_create(lv_screen_active());
    lv_obj_set_size(root, LV_PCT(100), LV_PCT(100));
    lv_obj_set_layout(root, LV_LAYOUT_GRID);
    static const int32_t col_dsc[] = {LV_GRID_FR(1), LV_GRID_FR(1), LV_GRID_TEMPLATE_LAST};
    static const int32_t row_dsc[] = {LV_GRID_FR(1), LV_GRID_CONTENT, LV_GRID_TEMPLATE_LAST};
    lv_obj_set_grid_col_dsc_array(root, col_dsc, row_dsc);
    container = lv_obj_create(root);
    lv_obj_remove_style_all(container);
    lv_obj_set_grid_cell(container, LV_GRID_ALIGN_STRETCH, 0, 2, LV_GRID_ALIGN_STRETCH, 0, 1);

    lv_obj_t * push_btn = lv_button_create(root);
    lv_obj_t * push_label = lv_label_create(push_btn);
    lv_label_set_text(push_label, "Push");

    lv_obj_t * pop_btn = lv_button_create(root);
    lv_obj_t * pop_label = lv_label_create(pop_btn);
    lv_label_set_text(pop_label, "Pop");
    lv_obj_set_grid_cell(pop_btn, LV_GRID_ALIGN_START, 0, 1, LV_GRID_ALIGN_CENTER, 1, 1);
    lv_obj_set_grid_cell(pop_btn, LV_GRID_ALIGN_END, 1, 1, LV_GRID_ALIGN_CENTER, 1, 1);

    lv_fragment_manager_t * manager = lv_fragment_manager_create(NULL);
    /* Clean up the fragment manager before objects in containers got deleted */
    lv_obj_add_event_cb(root, sample_container_delete, LV_EVENT_DELETE, manager);

    int depth = 0;
    lv_fragment_t * fragment = lv_fragment_create(&sample_cls, &depth);
    lv_fragment_manager_push(manager, fragment, &container);
    lv_obj_add_event_cb(push_btn, sample_push_click, LV_EVENT_CLICKED, manager);
    lv_obj_add_event_cb(pop_btn, sample_pop_click, LV_EVENT_CLICKED, manager);
}

static void sample_fragment_ctor(lv_fragment_t * self, void * args)
{
    LV_UNUSED(args);
    ((sample_fragment_t *) self)->depth = *((int *) args);
    ((sample_fragment_t *) self)->counter = 0;
}

static lv_obj_t * sample_fragment_create_obj(lv_fragment_t * self, lv_obj_t * parent)
{
    sample_fragment_t * fragment = (sample_fragment_t *) self;
    lv_obj_t * content = lv_obj_create(parent);
    lv_obj_remove_style_all(content);
    lv_obj_set_style_bg_opa(content, LV_OPA_50, 0);
    lv_obj_set_style_bg_color(content, lv_palette_main(LV_PALETTE_YELLOW), 0);
    lv_obj_set_size(content, LV_PCT(100), LV_PCT(100));
    lv_obj_set_flex_flow(content, LV_FLEX_FLOW_COLUMN);
    lv_obj_t * depth = lv_label_create(content);
    lv_label_set_text_fmt(depth, "Depth: %d", fragment->depth);
    lv_obj_t * label = lv_label_create(content);
    fragment->label = label;
    lv_label_set_text_fmt(label, "The button has been pressed %d times", fragment->counter);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_t * inc_btn = lv_button_create(content);
lv_obj_t * inc_label = lv_label_create(inc_btn);
lv_label_set_text(inc_label, "+1");
lv_obj_add_event_cb(inc_btn, sample_fragment_inc_click, LV_EVENT_CLICKED, fragment);

return content;
}

static void sample_push_click(lv_event_t * e)
{
    lv_fragment_manager_t * manager = (lv_fragment_manager_t *) lv_event_get_user_data(e);
    size_t stack_size = lv_fragment_manager_get_stack_size(manager);
    lv_fragment_t * fragment = lv_fragment_create(&sample_cls, &stack_size);
    lv_fragment_manager_push(manager, fragment, &container);
}

static void sample_pop_click(lv_event_t * e)
{
    lv_fragment_manager_t * manager = (lv_fragment_manager_t *) lv_event_get_user_data(e);
    lv_fragment_manager_pop(manager);
}

static void sample_container_delete(lv_event_t * e)
{
    lv_fragment_manager_t * manager = (lv_fragment_manager_t *) lv_event_get_user_data(e);
    lv_fragment_manager_delete(manager);
}

static void sample_fragment_inc_click(lv_event_t * e)
{
    sample_fragment_t * fragment = (sample_fragment_t *) lv_event_get_user_data(e);
    fragment->counter++;
    lv_label_set_text_fmt(fragment->label, "The button has been pressed %d times", fragment->counter);
}

#endif

```

### 9.3.3 API

*lv\_fragment.h*

*lv\_fragment\_private.h*

## 9.4 Grid navigation

Grid navigation (gridnav for short) is a feature that moves focus among a set of child Widgets via arrow-key presses.

If the child Widgets are arranged into a grid-like layout then the up, down, left and right arrows move focus to the nearest sibling in the respective direction.

It doesn't matter how the children are positioned, as only the current x and y coordinates are considered. This means that gridnav works with manually positioned children, as well as *Flex* and *Grid* layouts.

Gridnav also works if the children are arranged into a single row or column. This is useful, for example, to simplify navigation among items in a *List Widget*.

Gridnav assumes that the Widget to which gridnav is added is part of a *group*. This way, if the Widget with gridnav has focus, the arrow key presses are automatically forwarded to the Widget so that gridnav can process the arrow keys.

To move the focus to the next widget of the group use *LV\_KEY\_NEXT* or *LV\_KEY\_PREV*. Optionally you can also use *lv\_group\_focus\_next()* or *lv\_group\_focus\_prev()* or the TAB key on keyboard as usual.

If the container is scrollable and the focused child is out of the view, gridnav will automatically scroll the child into view.

### 9.4.1 Utilizzo

To add gridnav behavior to any Widget (e.g. one serving as a container for child Widgets that the end user will navigate among using arrow keys) use *lv\_gridnav\_add(container, flags)*.

The *flags* argument controls the navigation behavior:

- *LV\_GRIDNAV\_CTRL\_NONE*: Default settings
- *LV\_GRIDNAV\_CTRL\_ROLLOVER*: If there is no next/previous Widget in a direction, the focus goes to the Widget in the next/previous row (on left/right keys) or first/last row (on up/down keys)

- ***LV\_GRIDNAV\_CTRL\_SCROLL\_FIRST***: If an arrow is pressed and the focused Widget can be scrolled in that direction then it will be scrolled instead of going to the next/previous Widget. If there is no more room for scrolling the next/previous Widget will receive focus normally.
- ***LV\_GRIDNAV\_CTRL\_HORIZONTAL\_MOVE\_ONLY***: Only use the left/right keys for grid navigation. Up/down key events will be sent to the Widget that has focus.
- ***LV\_GRIDNAV\_CTRL\_VERTICAL\_MOVE\_ONLY***: Only use the up/down keys for grid navigation. Left/right key events will be sent to the Widget that has focus.

While the above behaviors can be combined by bit-wise OR-ing the above values together, ***LV\_GRIDNAV\_CTRL\_HORIZONTAL\_MOVE\_ONLY*** and ***LV\_GRIDNAV\_CTRL\_VERTICAL\_MOVE\_ONLY*** should not be used together.

`lv_gridnav_remove(container)` Removes gridnav behavior from a Widget.

## 9.4.2 Focusable Widgets

A Widget needs to be clickable or click focusable (***LV\_OBJ\_FLAG\_CLICKABLE*** or ***LV\_OBJ\_FLAG\_CLICK\_FOCUSABLE***) and not hidden (***LV\_OBJ\_FLAG\_HIDDEN***) to receive focus via gridnav.

## 9.4.3 Esempi

### Basic grid navigation

```
#include "../../lv_examples.h"
#if LV_USE_GRIDNAV && LV_USE_FLEX && LV_BUILD_EXAMPLES

/***
 * Demonstrate a basic grid navigation
 */
void lv_example_gridnav_1(void)
{
    /*It's assumed that the default group is set and
     *there is a keyboard indev*/
    lv_obj_t * cont1 = lv_obj_create(lv_screen_active());
    lv_gridnav_add(cont1, LV_GRIDNAV_CTRL_NONE);

    /*Use flex here, but works with grid or manually placed objects as well*/
    lv_obj_set_flex_flow(cont1, LV_FLEX_FLOW_ROW_WRAP);
    lv_obj_set_style_bg_color(cont1, lv_palette_lighten(LV_PALETTE_BLUE, 5), LV_STATE_FOCUSED);
    lv_obj_set_size(cont1, lv_pct(50), lv_pct(100));

    /*Only the container needs to be in a group*/
    lv_group_add_obj(lv_group_get_default(), cont1);

    lv_obj_t * label = lv_label_create(cont1);
    lv_label_set_text_fmt(label, "No rollover");

    uint32_t i;
    for(i = 0; i < 10; i++) {
        lv_obj_t * obj = lv_button_create(cont1);
        lv_obj_set_size(obj, 70, LV_SIZE_CONTENT);
        lv_obj_add_flag(obj, LV_OBJ_FLAG_CHECKABLE);
        lv_group_remove_obj(obj); /*Not needed, we use the gridnav instead*/

        label = lv_label_create(obj);
        lv_label_set_text_fmt(label, "%" LV_PRIu32 "", i);
        lv_obj_center(label);
    }

    /* Create a second container with rollover grid nav mode.*/
    lv_obj_t * cont2 = lv_obj_create(lv_screen_active());
    lv_gridnav_add(cont2, LV_GRIDNAV_CTRL_ROLLOVER);
    lv_obj_set_style_bg_color(cont2, lv_palette_lighten(LV_PALETTE_BLUE, 5), LV_STATE_FOCUSED);
    lv_obj_set_size(cont2, lv_pct(50), lv_pct(100));
    lv_obj_align(cont2, LV_ALIGN_RIGHT_MID, 0, 0);

    label = lv_label_create(cont2);
    lv_obj_set_width(label, lv_pct(100));
    lv_label_set_text_fmt(label, "Rollover\nUse tab to focus the other container");

    /*Only the container needs to be in a group*/
    lv_group_add_obj(lv_group_get_default(), cont2);

    /*Add and place some children manually*/
    lv_obj_t * ta = lv_textarea_create(cont2);
    lv_obj_set_size(ta, lv_pct(100), 80);
    lv_obj_set_pos(ta, 0, 80);
    lv_group_remove_obj(ta); /*Not needed, we use the gridnav instead*/

    lv_obj_t * cb = lv_checkbox_create(cont2);
    lv_obj_set_pos(cb, 0, 170);
    lv_group_remove_obj(cb); /*Not needed, we use the gridnav instead/

    lv_obj_t * swl = lv_switch_create(cont2);
    lv_obj_set_pos(swl, 0, 200);
    lv_group_remove_obj(swl); /*Not needed, we use the gridnav instead*/

    lv_obj_t * sw2 = lv_switch_create(cont2);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_pos(sw2, lv_pct(50), 200);
lv_group_remove_obj(sw2); /*Not needed, we use the gridnav instead*/
}

#endif

```

## Grid navigation on a list

```

#include "../../lv_examples.h"
#if LV_USE_GRIDNAV && LV_USE_LIST && LV_BUILD_EXAMPLES

/**
 * Grid navigation on a list
 */
void lv_example_gridnav_2(void)
{
    /*It's assumed that the default group is set and
     *there is a keyboard indev*/

    lv_obj_t * list1 = lv_list_create(lv_screen_active());
    lv_gridnav_add(list1, LV_GRIDNAV_CTRL_NONE);
    lv_obj_set_size(list1, lv_pct(45), lv_pct(80));
    lv_obj_align(list1, LV_ALIGN_LEFT_MID, 5, 0);
    lv_obj_set_style_bg_color(list1, lv_palette_lighten(LV_PALETTE_BLUE, 5), LV_STATE_FOCUSED);
    lv_group_add_obj(lv_group_get_default(), list1);

    char buf[32];
    uint32_t i;
    for(i = 0; i < 15; i++) {
        lv_snprintf(buf, sizeof(buf), "File %d", i + 1);
        lv_obj_t * item = lv_list_add_button(list1, LV_SYMBOL_FILE, buf);
        lv_obj_set_style_bg_opa(item, 0, 0);
        lv_group_remove_obj(item); /*Not needed, we use the gridnav instead*/
    }

    lv_obj_t * list2 = lv_list_create(lv_screen_active());
    lv_gridnav_add(list2, LV_GRIDNAV_CTRL_ROLLOVER);
    lv_obj_set_size(list2, lv_pct(45), lv_pct(80));
    lv_obj_align(list2, LV_ALIGN_RIGHT_MID, -5, 0);
    lv_obj_set_style_bg_color(list2, lv_palette_lighten(LV_PALETTE_BLUE, 5), LV_STATE_FOCUSED);
    lv_group_add_obj(lv_group_get_default(), list2);

    for(i = 0; i < 15; i++) {
        lv_snprintf(buf, sizeof(buf), "Folder %d", i + 1);
        lv_obj_t * item = lv_list_add_button(list2, LV_SYMBOL_DIRECTORY, buf);
        lv_obj_set_style_bg_opa(item, 0, 0);
        lv_group_remove_obj(item);
    }
}
#endif

```

## Nested grid navigations

```

#include "../../lv_examples.h"
#if LV_USE_GRIDNAV && LV_USE_FLEX && LV_BUILD_EXAMPLES

static void cont_sub_event_cb(lv_event_t * e)
{
    uint32_t k = lv_event_get_key(e);
    lv_obj_t * obj = lv_event_get_target_obj(e);
    if(k == LV_KEY_ENTER) {
        lv_group_focus_obj(obj);
    }
    else if(k == LV_KEY_ESC) {
        lv_group_focus_next(lv_obj_get_group(obj));
    }
}

/**
 * Nested grid navigations
 */
void lv_example_gridnav_3(void)
{
    /*It's assumed that the default group is set and
     *there is a keyboard indev*/

    lv_obj_t * cont_main = lv_obj_create(lv_screen_active());
    lv_gridnav_add(cont_main, (LV_GRIDNAV_CTRL_ROLLOVER | LV_GRIDNAV_CTRL_SCROLL_FIRST));

    /*Only the container needs to be in a group/
    lv_group_add_obj(lv_group_get_default(), cont_main);

    /*Use flex here, but works with grid or manually placed objects as well/
    lv_obj_set_flex_flow(cont_main, LV_FLEX_FLOW_ROW_WRAP);
    lv_obj_set_style_bg_color(cont_main, lv_palette_lighten(LV_PALETTE_BLUE, 5), LV_STATE_FOCUSED);
    lv_obj_set_size(cont_main, lv_pct(80), LV_SIZE_CONTENT);

    lv_obj_t * btn;
    lv_obj_t * label;

    btn = lv_button_create(cont_main);
    lv_group_remove_obj(btn);
    label = lv_label_create(btn);
    lv_label_set_text(label, "Button 1");

    btn = lv_button_create(cont_main);
    lv_group_remove_obj(btn);
    label = lv_label_create(btn);
    lv_label_set_text(label, "Button 2");

    /*Create another container with long text to show how LV_GRIDNAV_CTRL_SCROLL_FIRST works/
    lv_obj_t * cont_sub1 = lv_obj_create(cont_main);

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_set_size(cont_sub1, lv_pct(100), 100);

label = lv_label_create(cont_sub1);
lv_obj_set_style_bg_color(cont_sub1, lv_palette_lighten(LV_PALETTE_RED, 5), LV_STATE_FOCUSED);
lv_obj_set_width(label, lv_pct(100));
lv_label_set_text(label,
    "I'm a very long text which is makes my container scrollable. "
    "As LV_GRIDNAV_FLAG_SCROLL_FIRST is enabled arrow will scroll me first "
    "and a new objects will be focused only when an edge is reached with the scrolling.\n\n"
    "This is only some placeholder text to be sure the parent will be scrollable. \n\n"
    "Hello world!\n"
    "Hello world!\n"
    "Hello world!\n"
    "Hello world!\n"
    "Hello world!\n"
    "Hello world!\n"
    "Hello world!");
}

/*Create a third container that can be focused with ENTER and contains another grid nav*/
lv_obj_t * cont_sub2 = lv_obj_create(cont_main);
lv_gridnav_add(cont_sub2, LV_GRIDNAV_CTRL_ROLLOVER);
/*Only the container needs to be in a group*/
lv_group_add_obj(lv_group_get_default(), cont_sub2);

lv_obj_add_event_cb(cont_sub2, cont_sub_event_cb, LV_EVENT_KEY, NULL);

/*Use flex here, but works with grid or manually placed objects as well*/
lv_obj_set_flex_flow(cont_sub2, LV_FLEX_FLOW_ROW_WRAP);
lv_obj_set_style_bg_color(cont_sub2, lv_palette_lighten(LV_PALETTE_RED, 5), LV_STATE_FOCUSED);
lv_obj_set_size(cont_sub2, lv_pct(100), LV_SIZE_CONTENT);

label = lv_label_create(cont_sub2);
lv_label_set_text(label, "Use ENTER/ESC to focus/defocus this container");
lv_obj_set_width(label, lv_pct(100));

btn = lv_button_create(cont_sub2);
lv_group_remove_obj(btn);
label = lv_label_create(btn);
lv_label_set_text(label, "Button 3");

btn = lv_button_create(cont_sub2);
lv_group_remove_obj(btn);
label = lv_label_create(btn);
lv_label_set_text(label, "Button 4");

}

#endif

```

## Simple navigation on a list widget

```

#include "../../lv_examples.h"
#if LV_USE_GRIDNAV && LV_USE_FLEX && LV_BUILD_EXAMPLES

static void event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target_obj(e);
    lv_obj_t * list = lv_obj_get_parent(obj);
    LV_UNUSED(list); /*If logging is disabled*/
    LV_LOG_USER("Clicked: %s", lv_list_get_button_text(list, obj));
}

/**
 * Simple navigation on a list widget
 */
void lv_example_gridnav_4(void)
{
    /*It's assumed that the default group is set and
     *there is a keyboard indev*/
    lv_obj_t * list = lv_list_create(lv_screen_active());
    lv_gridnav_add(list, LV_GRIDNAV_CTRL_ROLLOVER);
    lv_obj_align(list, LV_ALIGN_LEFT_MID, 10, 0);
    lv_group_add_obj(lv_group_get_default(), list);

    uint32_t i;
    for(i = 0; i < 20; i++) {
        char buf[32];

        /*Add some separators too, they are not focusable by gridnav*/
        if((i % 5) == 0) {
            lv_snprintf(buf, sizeof(buf), "Section %d", i / 5 + 1);
            lv_list_add_text(list, buf);
        }

        lv_snprintf(buf, sizeof(buf), "File %d", i + 1);
        lv_obj_t * item = lv_list_add_button(list, LV_SYMBOL_FILE, buf);
        lv_obj_add_event_cb(item, event_handler, LV_EVENT_CLICKED, NULL);
        lv_group_remove_obj(item); /*The default group adds it automatically*/
    }

    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_align(btn, LV_ALIGN_RIGHT_MID, -10, 0);
    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Button");
}
#endif

```

## Grid navigation for only one axis

```
#include "../../lv_examples.h"
#if LV_USE_GRIDNAV && LV_USE_FLEX && LV_BUILD_EXAMPLES

static const char * opts[] = {"0\n1\n2\n3\n4\n5", "0\n1\n2\n3\n4\n5\n6\n7\n8\n9", "s\nm\nh"};
static const int32_t opts_counts[] = {6, 10, 3};

static lv_obj_t * sliders[3];
static lv_obj_t * rollers[3];

static void slider_key_cb(lv_event_t * e)
{
    uint8_t i = (uint32_t)(uintptr_t)lv_event_get_user_data(e);
    lv_roller_set_selected(rollers[i], lv_slider_get_value(sliders[i]), LV_ANIM_ON);
}
static void roller_key_cb(lv_event_t * e)
{
    uint8_t i = (uint32_t)(uintptr_t)lv_event_get_user_data(e);
    lv_slider_set_value(sliders[i], lv_roller_get_selected(rollers[i]), LV_ANIM_ON);
}

/**
 * Grid navigation for only one axis
 */
void lv_example_gridnav_5(void)
{
    /*It's assumed that the default group is set and
     *there is a keyboard indev*/
    lv_group_t * group = lv_group_get_default();
    lv_obj_t * cont;

    cont = lv_obj_create(lv_screen_active());
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(cont, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj_set_size(cont, lv_pct(100), lv_pct(50));
    lv_obj_align(cont, LV_ALIGN_TOP_MID, 0, 0);
    /* only up/down keys will be used for grid navigation in this container. */
    /* left/right will be sent to the sliders */
    lv_gridnav_add(cont, LV_GRIDNAV_CTRL_VERTICAL_MOVE_ONLY);
    lv_group_add_obj(group, cont);
    for(uint32_t i = 0; i < 3; i++) {
        lv_obj_t * slider = lv_slider_create(cont);
        lv_slider_set_range(slider, 0, opts_counts[i] - 1);
        lv_group_remove_obj(slider);
        lv_obj_set_width(slider, lv_pct(85));
        sliders[i] = slider;
        lv_obj_add_event_cb(slider, slider_key_cb, LV_EVENT_KEY, (void*)(uintptr_t)i);
    }

    cont = lv_obj_create(lv_screen_active());
    lv_obj_set_flex_flow(cont, LV_FLEX_FLOW_ROW);
    lv_obj_set_flex_align(cont, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj_set_size(cont, lv_pct(100), lv_pct(50));
    lv_obj_align(cont, LV_ALIGN_BOTTOM_MID, 0, 0);
    /* only left/right keys will be used for grid navigation in this container. */
    /* up/down will be sent to the rollers */
    lv_gridnav_add(cont, LV_GRIDNAV_CTRL_HORIZONTAL_MOVE_ONLY);
    lv_group_add_obj(group, cont);
    for(uint32_t i = 0; i < 3; i++) {
        lv_obj_t * roller = lv_roller_create(cont);
        lv_roller_set_options(roller, opts[i], LV_ROLLER_MODE_INFINITE);
        lv_obj_set_size(roller, lv_pct(30), lv_pct(100));
        lv_group_remove_obj(roller);
        rollers[i] = roller;
        lv_obj_add_event_cb(roller, roller_key_cb, LV_EVENT_KEY, (void*)(uintptr_t)i);
    }
}

#endif
```

## 9.4.4 API

*lv\_gridnav.h*

## 9.5 Pinyin IME

Pinyin IME provides an API to provide Chinese Pinyin input method (Chinese input) for a Keyboard Widget, which supports both 26-key and 9-key input modes. You can think of `lv ime pinyin` as a Pinyin input method plug-in for the Keyboard Widget.

Normally, an environment where `Keyboard (lv_keyboard)` can run can also run `lv ime pinyin`. There are two main influencing factors: the size of the font file and the size of the dictionary.

### 9.5.1 Utilizzo

Enable `LV_USEIME_PINYIN` in `lv_conf.h`.

First use `lv_ime_pinyin_create(lv_screen_active())` to create a Pinyin input-method plug-in, then use `lv_ime_pinyin_set_keyboard(pinyin_ime, kb)` to add the Keyboard Widget you created to the Pinyin input method plug-in. You can use `lv_ime_pinyin_set_dict(pinyin_ime,`

`your_dict`) to use a custom dictionary. If you don't want to use the built-in dictionary, you can disable `LV_IME_PINYIN_USE_DEFAULT_DICT` in `lv_conf.h`, which can save a lot of memory space.

The built-in thesaurus is customized based on the `LV_FONT_SOURCE_HAN_SANS_SC_16_CJK` font library, which currently has more than 1,000 of the most common CJK radicals, so it is recommended to use a custom font and thesaurus.

In the process of using the Pinyin input method plug-in, you can change the Keyboard and dictionary at any time.

### 9.5.2 Custom Dictionary

If you don't want to use the built-in Pinyin dictionary, or if you feel that the built-in phonetic dictionary consumes too much memory, you can use a custom dictionary.

Customizing the dictionary is very simple.

First, set `LV_IME_PINYIN_USE_DEFAULT_DICT` to 0 in `lv_conf.h`

Then, write a dictionary in the following format.

#### Dictionary format

The arrangement order of each pinyin syllable is very important. If you need to customize your own thesaurus according to the Hanyu Pinyin syllable table, you can read [here](#) to learn about the Hanyu Pinyin syllables and the syllable table.

Then, write your own dictionary according to the following format:

```
lv_100ask_pinyin_dict_t your_pinyin_dict[] = {
    { "a", "ㄞㄞㄞ" },
    { "ai", "ㄞㄞㄞㄞㄞ" },
    { "an", "ㄞㄞㄞㄞㄞㄞ" },
    { "ang", "ㄞㄞㄞㄞㄞㄞ" },
    { "ao", "ㄞㄞㄞㄞ" },
    { "ba", "ㄞㄞㄞㄞㄞ" },
    { "bai", "ㄞㄞㄞㄞㄞ" },
    /* ..... */
    { "zuo", "ㄞㄞㄞㄞㄞ" },
    {NULL, NULL}
```

The last item must be `{null, null}`, or it will not work properly.

#### Applying a new dictionary

After writing a dictionary according to the above dictionary format, you only need to call this function to set up and use your dictionary:

```
lv_obj_t * pinyin_ime = lv_100ask_pinyin_ime_create(lv_screen_active());
lv_100ask_pinyin_ime_set_dict(pinyin_ime, your_pinyin_dict);
```

### 9.5.3 Modes

`lv_ime_pinyin` has the following modes:

- `LV_IME_PINYIN_MODE_K26`: Pinyin 26-key input mode
- `LV_IME_PINYIN_MODE_K9`: Pinyin 9-key input mode
- `LV_IME_PINYIN_MODE_K9_NUMBER`: Numeric keypad mode

The Keyboard's TEXT-mode layout contains buttons to change mode.

To set the mode manually, use `lv_ime_pinyin_set_mode(pinyin_ime, mode)`. The default mode is `LV_IME_PINYIN_MODE_K26`.

### 9.5.4 Esempio

#### Pinyin IME 26 key input

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_USE_TEXTAREA && LV_FONT_SOURCE_HAN_SANS_SC_16_CJK && LV_USEIME_PINYIN && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target_obj(e);
    lv_obj_t * kb = (lv_obj_t *)lv_event_get_user_data(e);

    if(code == LV_EVENT_FOCUSED) {
        if(lv_indev_get_type(lv_indev_active()) != LV_INDEV_TYPE_KEYPAD) {
            lv_keyboard_set_textarea(kb, ta);
            lv_obj_remove_flag(kb, LV_OBJ_FLAG_HIDDEN);
        }
    }
    else if(code == LV_EVENT_CANCEL) {
        lv_obj_add_flag(kb, LV_OBJ_FLAG_HIDDEN);
        lv_obj_remove_state(ta, LV_STATE_FOCUSED);
        lv_indev_reset(NULL, ta); /*To forget the last clicked object to make it focusable again*/
    }
}

void lv_example_ime_pinyin_1(void)
{
    lv_obj_t * pinyin_ime = lv_ime_pinyin_create(lv_screen_active());
    lv_obj_set_style_text_font(pinyin_ime, &lv_font_source_han_sans_sc_16_cjk, 0);
    //lv_ime_pinyin_set_dict(pinyin_ime, your_dict); // Use a custom dictionary. If it is not set, the built-in dictionary will be used.

    /* tal */
    lv_obj_t * tal = lv_textarea_create(lv_screen_active());
    lv_textarea_set_one_line(tal, true);
    lv_obj_set_style_text_font(tal, &lv_font_source_han_sans_sc_16_cjk, 0);
    lv_obj_align(tal, LV_ALIGN_TOP_LEFT, 0, 0);

    /*Create a keyboard and add it to ime_pinyin*/
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());
    lv_ime_pinyin_set_keyboard(pinyin_ime, kb);
    lv_keyboard_set_textarea(kb, tal);

    lv_obj_add_event_cb(tal, ta_event_cb, LV_EVENT_ALL, kb);

    /*Get the cand_panel, and adjust its size and position*/
    lv_obj_t * cand_panel = lv_ime_pinyin_get_cand_panel(pinyin_ime);
    lv_obj_set_size(cand_panel, LV_PCT(100), LV_PCT(10));
    lv_obj_align_to(cand_panel, kb, LV_ALIGN_OUT_TOP_MID, 0, 0);

    /*Try using ime_pinyin to output the Chinese below in the tal above*/
    lv_obj_t * cz_label = lv_label_create(lv_screen_active());
    lv_label_set_text(cz_label, "中文字体嵌入式系统\n");
    lv_obj_set_style_text_font(cz_label, &lv_font_source_han_sans_sc_16_cjk, 0);
    lv_obj_set_width(cz_label, 310);
    lv_obj_align_to(cz_label, tal, LV_ALIGN_OUT_BOTTOM_LEFT, 0, 0);
}
#endif
```

## Pinyin IME 9 key input

```
#include "../../lv_examples.h"
#if LV_USE_LABEL && LV_USE_TEXTAREA && LV_FONT_SOURCE_HAN_SANS_SC_16_CJK && LV_USEIME_PINYIN && LV_IME_PINYIN_USE_K9_MODE && LV_BUILD_EXAMPLES

static void ta_event_cb(lv_event_t * e)
{
    lv_event_code_t code = lv_event_get_code(e);
    lv_obj_t * ta = lv_event_get_target_obj(e);
    lv_obj_t * kb = (lv_obj_t *)lv_event_get_user_data(e);

    if(code == LV_EVENT_FOCUSED) {
        if(lv_indev_get_type(lv_indev_active()) != LV_INDEV_TYPE_KEYPAD) {
            lv_keyboard_set_textarea(kb, ta);
            lv_obj_remove_flag(kb, LV_OBJ_FLAG_HIDDEN);
        }
    }
    else if(code == LV_EVENT_READY) {
        lv_obj_add_flag(kb, LV_OBJ_FLAG_HIDDEN);
        lv_obj_remove_state(ta, LV_STATE_FOCUSED);
        lv_indev_reset(NULL, ta); /*To forget the last clicked object to make it focusable again*/
    }
}

void lv_example_ime_pinyin_2(void)
{
    lv_obj_t * pinyin_ime = lv_ime_pinyin_create(lv_screen_active());
    lv_obj_set_style_text_font(pinyin_ime, &lv_font_source_han_sans_sc_16_cjk, 0);
    //lv_ime_pinyin_set_dict(pinyin_ime, your_dict); // Use a custom dictionary. If it is not set, the built-in dictionary will be used.

    /* tal */
    lv_obj_t * tal = lv_textarea_create(lv_screen_active());
    lv_textarea_set_one_line(tal, true);
    lv_obj_set_style_text_font(tal, &lv_font_source_han_sans_sc_16_cjk, 0);
    lv_obj_align(tal, LV_ALIGN_TOP_LEFT, 0, 0);

    /*Create a keyboard and add it to ime_pinyin*/
    lv_obj_t * kb = lv_keyboard_create(lv_screen_active());
    lv_keyboard_set_textarea(kb, tal);

    lv_ime_pinyin_set_keyboard(pinyin_ime, kb);
    lv_ime_pinyin_set_mode(pinyin_ime,
        LV_IME_PINYIN_MODE_K9); // Set to 9-key input mode. Default: 26-key input(k26) mode.
    lv_obj_add_event_cb(tal, ta_event_cb, LV_EVENT_ALL, kb);

    /*Get the cand_panel, and adjust its size and position*/
    lv_obj_t * cand_panel = lv_ime_pinyin_get_cand_panel(pinyin_ime);
    lv_obj_set_size(cand_panel, LV_PCT(100), LV_PCT(10));
    lv_obj_align_to(cand_panel, kb, LV_ALIGN_OUT_TOP_MID, 0, 0);

    /*Try using ime_pinyin to output the Chinese below in the tal above*/
    lv_obj_t * cz_label = lv_label_create(lv_screen_active());
    lv_label_set_text(cz_label,
```

(continues on next page)

(continua dalla pagina precedente)

```

    "呵呵呵呵Embedded System呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵呵";
lv_obj_set_style_text_font(cz_label, &lv_font_source_han_sans_sc_16_cjk, 0);
lv_obj_set_width(cz_label, 310);
lv_obj_align_to(cz_label, tal, LV_ALIGN_OUT_BOTTOM_LEFT, 0, 0);
}
#endif

```

## 9.5.5 API

*lv\_ime\_pinyin.h*  
*lv\_ime\_pinyin\_private.h*

## 9.6 Image font

Draw image in **label** or **span** obj with **lv\_imgfont**. This is often used to display Unicode emoji icons in text.

Supported image formats: determined by enabled LVGL *image decoders*.

### 9.6.1 Utilizzo

Enable **LV\_USE\_IMGFONT** in **lv\_conf.h**.

To create a new *imgfont*:

```

static lv_font_t * imgfont;
...
imgfont = lv_imgfont_create(height, path_cb, user_data);

```

- **height** Font size.
- **path\_cb** A function to get the image path of a character. Pass **NULL** if no image should be shown, but the character itself.
- **user\_data** Pointer to user data.

To use the *imgfont* in a label, reference it: **lv\_obj\_set\_style\_text\_font(label, imgfont, LV\_PART\_MAIN)**

To destroy the *imgfont* that is no longer used, use **lv\_imgfont\_destroy(imgfont)**.

### 9.6.2 Esempio

Use emojis in a text.

```

#include "../../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_IMGFONT

static const void * get_imgfont_path(const lv_font_t * font, uint32_t unicode, uint32_t unicode_next,
                                    int32_t * offset_y, void * user_data)
{
    LV_UNUSED(font);
    LV_UNUSED(unicode_next);
    LV_UNUSED(offset_y);
    LV_UNUSED(user_data);

    LV_IMAGE_DECLARE(emoji_F617);

    if(unicode < 0xF000) return NULL;

    if(unicode == 0xF617) {
        return &emoji_F617;
    }
    else if(unicode == 0xF600) {
#if LV_USE_FFMPEG
        return "lvgl/examples/assets/emoji/F600.png";
#else
        return "A:lvgl/examples/assets/emoji/F600.png";
#endif
    }
    return NULL;
}

/***
 * draw img in label or span obj
 */
void lv_example_imgfont_1(void)

```

(continues on next page)

(continua dalla pagina precedente)

```
{
    lv_font_t * imgfont = lv_imgfont_create(80, get_imgfont_path, NULL);
    if(imgfont == NULL) {
        LV_LOG_ERROR("imgfont init error");
        return;
    }

    imgfont->fallback = LV_FONT_DEFAULT;

    lv_obj_t * label1 = lv_label_create(lv_screen_active());
    lv_label_set_text(label1, "12\uF600\uF617AB");
    lv_obj_set_style_text_font(label1, imgfont, LV_PART_MAIN);
    lv_obj_center(label1);
}
#else

void lv_example_imgfont_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "imgfont is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

### 9.6.3 API

*lv\_imgfont.h*

## 9.7 Monkey

The Monkey module provides LVGL applications with a simple monkey test. Monkey Testing is a technique where the user tests the application or system by providing random inputs and checking the behavior or seeing whether the application or system will crash. This module provides this service as simulated random input to stress test an LVGL application.

### 9.7.1 Utilizzo

First, enable *LV\_USE\_MONKEY* in *lv\_conf.h*.

Next, declare a variable (it can be local) of type *lv\_monkey\_config\_t* to define the configuration structure, initialize it using *lv\_monkey\_config\_init(cfg)* then set its *type* member to the desired type of *input device*, and set the *min* and *max* values for its *period\_range* and *input\_range* members to set the time ranges (in milliseconds) and input ranges the Monkey module will use to generate random input at random times.

Next, call *lv\_monkey\_create(cfg)* to create the Monkey. It returns a pointer to the *lv\_monkey\_t* created.

Finally call *lv\_monkey\_set\_enable(monkey, true)* to enable Monkey.

If you want to pause the monkey, call *lv\_monkey\_set\_enable(monkey, false)*. To delete it, call *lv\_monkey\_delete(monkey)*.

Note that *input\_range* has different meanings depending on the *type* input device:

- *LV\_INDEV\_TYPE\_POINTER*: No effect, click randomly within the pixels of the screen resolution.
- *LV\_INDEV\_TYPE\_ENCODER*: The minimum and maximum values of *enc\_diff*.
- *LV\_INDEV\_TYPE\_BUTTON*: The minimum and maximum values of *btn\_id*. Use *lv\_monkey\_get\_indev()* to get the input device, and use *lv\_indev\_set\_button\_points()* to map the key ID to the coordinates.
- *LV\_INDEV\_TYPE\_KEYPAD*: No effect, Send random *Tasti*.

### 9.7.2 Esempio

#### Touchpad monkey example

```
#include "../../lv_examples.h"
#if LV_USE_MONKEY && LV_BUILD_EXAMPLES

void lv_example_monkey_1(void)
{
    /*Create pointer monkey test*/
```

(continues on next page)

(continua dalla pagina precedente)

```

lv_monkey_config_t config;
lv_monkey_config_init(&config);
config.type = LV_INDEV_TYPE_POINTER;
config.period_range.min = 10;
config.period_range.max = 100;
lv_monkey_t * monkey = lv_monkey_create(&config);

/*Start monkey test*/
lv_monkey_set_enable(monkey, true);
}

#endif

```

## Encoder monkey example

```

#include "../../../lv_examples.h"
#if LV_USE_MONKEY && LV_BUILD_EXAMPLES

void lv_example_monkey_2(void)
{
    /*Create encoder monkey test*/
    lv_monkey_config_t config;
    lv_monkey_config_init(&config);
    config.type = LV_INDEV_TYPE_ENCODER;
    config.period_range.min = 50;
    config.period_range.max = 500;
    config.input_range.min = -5;
    config.input_range.max = 5;
    lv_monkey_t * monkey = lv_monkey_create(&config);

    /*Set the default group*/
    lv_group_t * group = lv_group_create();
    lv_indev_set_group(lv_monkey_get_indev(monkey), group);
    lv_group_set_default(group);

    /*Start monkey test*/
    lv_monkey_set_enable(monkey, true);
}
#endif

```

## Button monkey example

```

#include "../../../lv_examples.h"
#if LV_USE_MONKEY && LV_BUILD_EXAMPLES

void lv_example_monkey_3(void)
{
    static lv_point_t btn_points[3];
    int32_t hor_res = LV_HOR_RES;

    /*Create button monkey test*/
    lv_monkey_config_t config;
    lv_monkey_config_init(&config);
    config.type = LV_INDEV_TYPE_BUTTON;
    config.period_range.min = 50;
    config.period_range.max = 500;
    config.input_range.min = 0;
    config.input_range.max = sizeof(btn_points) / sizeof(lv_point_t) - 1;
    lv_monkey_t * monkey = lv_monkey_create(&config);

    /*Set the coordinates bound to the button*/
    btn_points[0].x = hor_res / 4;
    btn_points[0].y = 10;
    btn_points[1].x = hor_res / 2;
    btn_points[1].y = 10;
    btn_points[2].x = hor_res * 3 / 4;
    btn_points[2].y = 10;

    lv_indev_set_button_points(lv_monkey_get_indev(monkey), btn_points);

    /*Start monkey test*/
    lv_monkey_set_enable(monkey, true);
}
#endif

```

## 9.7.3 API

*lv\_monkey.h*

## 9.8 Widget ID

Widgets can optionally have identifiers added to their functionality if needed for the application. Exactly how that happens is designed to be flexible, and can morph with the needs of the application. It can even be a timestamp or other data current at the time the Widget was created.

### 9.8.1 Utilizzo

Enable Widget ID functionality by setting `LV_USE_OBJ_ID` to 1 in `lv_conf.h`.

Once enabled, several things change:

- each Widget will now have a `void * id` field called `id`;
- these two API functions become available:
  - `lv_obj_get_id(widget)`,
  - `lv_obj_find_by_id(widget, id)`;
- several more Widget-ID-related API functions become available if `LV_USE_OBJ_ID_BUILTIN` is non-zero (more on this below);
- two additional configuration macros both `LV_OBJ_ID_AUTO_ASSIGN` and `LV_USE_OBJ_ID_BUILTIN` now have meaning.

#### `LV_OBJ_ID_AUTO_ASSIGN`

This macro in `lv_conf.h` defaults to whatever value `LV_USE_OBJ_ID` equates to. You can change this if you wish. Either way, if it equates to a non-zero value, it causes two things to happen:

- `lv_obj_assign_id(class_p, widget)` will be called at the end of each Widget's creation, and
- `lv_obj_free_id(widget)` will be called at the end of the sequence when each Widget is deleted.

Because of this timing, custom versions of these functions can be used according to the below, and they can even be used like "event hooks" to implement a trace operation that occurs when each Widget is created and deleted.

#### `lv_obj_assign_id(class_p, widget)`

This function (whether provided by LVGL or by you --- more on this below) is responsible for assigning a value to the Widget's `id` field, and possibly do other things, depending on the implementation.

#### `lv_obj_free_id(widget)`

This function (whether provided by LVGL or by you --- more on this below) is responsible for doing the clean-up of any resources allocated by `lv_obj_assign_id()`

#### `LV_USE_OBJ_ID_BUILTIN`

This macro in `lv_conf.h` equates to 1 by default. You can change this if you wish. When it equates to a non-zero value the following function implementations are provided by LVGL:

- `lv_obj_assign_id(class_p, widget)`
- `lv_obj_free_id(widget)`
- `lv_obj_set_id(widget, id)`
- `lv_obj_stringify_id(widget, buf, len)`
- `lv_obj_id_compare(id1, id2)`

These supply the default implementation for Widget IDs, namely that for each Widget created, `lv_obj_stringify_id(widget, buf, len)` will produce a unique string for it. Example: if the following 6 Widgets are created in this sequence:

- Screen
- Etichetta
- Pulsante
- Etichetta

- Etichetta
- Immagine

the strings produced by `lv_obj_stringify_id(widget, buf, len)` would be

- obj1
- label1
- btn1
- label2
- label3
- image1

rispettivamente.

### Using a custom ID generator

If you wish, you can provide custom implementations for several Widget-ID related functions. You do this by first setting `LV_USE_OBJ_ID_BUILTIN` to 0 in `lv_conf.h`.

You will then need to provide implementations for the following functions (and link them with LVGL):

```
const char * lv_obj_stringify_id(lv_obj_t * widget, char * buf, uint32_t len);
int lv_obj_id_compare(const void * id1, const void * id2);
```

If `LV_OBJ_ID_AUTO_ASSIGN` equates to a non-zero value (or if you otherwise simply need to use them), you will also need to provide implementations for:

```
void lv_obj_assign_id(const lv_obj_class_t * class_p, lv_obj_t * widget);
void lv_obj_free_id(lv_obj_t * widget);
```

If `LV_BUILD_TEST` equates to a non-zero value and you are including LVGL test code in your compile (or if you otherwise simply need to use them), you will also need to provide an implementation for:

```
void lv_obj_set_id(lv_obj_t * widget, void * id);
```

Examples of implementations of these functions exist in `lv_obj_id_builtin.c`, but you are free to use a different design if needed.

`lv_obj_stringify_id()` converts the passed `widget` to a string representation (typically incorporating the `id` field) and writes it into the buffer provided in its `buf` argument.

`lv_obj_id_compare()` compares 2 `void * id` values and returns 0 when they are considered equal, and non-zero otherwise.

If `LV_OBJ_ID_AUTO_ASSIGN` equates to a non-zero value, `lv_obj_assign_id()` is called when a Widget is created. It is responsible for assigning a value to the Widget's `id` field. A pointer to the Widget's final class is passed in its `class_p` argument in case it is needed for determining the value for the `id` field, or for other possible needs related to your design for Widget IDs. Note that this pointer may be different than `widget->class_p` which is the class of the Widget currently being created.

If `LV_OBJ_ID_AUTO_ASSIGN` equates to a non-zero value, `lv_obj_free_id()` is called when a Widget is deleted. It needs to perform the clean-up for any resources allocated by `lv_obj_assign_id()`.

### Dumping a Widget Tree

Regardless of the state of any of the above macros, the function `lv_obj_dump_tree(widget)` provides a "dump" of the Widget Tree for the specified Widget (that Widget plus all its children recursively) using the currently-configured method used by the `LV_LOG_USER` macro. If NULL is passed instead of a pointer to a "root" Widget, the dump will include the Widget Tree for all `Screen`, for all `Displays` in the system.

For `LV_LOG_USER` to produce output, the following needs to be true in `lv_conf.h`:

- `LV_USE_LOG` must equate to a non-zero value

- `LV_LOG_LEVEL <= LV_LOG_LEVEL_USER`

It will recursively walk through all that Widget's children (starting with the Widget itself) and print the Widget's parent's address, the Widget's address, and if `LV_USE_OBJ_ID` equates to a non-zero value, will also print the output of `lv_obj_stringify_id()` for that Widget.

This can be useful in the event of a UI crash. From that log you can examine the state of the Widget Tree when `lv_obj_dump_tree(widget)` was called.

For example, if a pointer to a deleted Widget is stored in a Timer's `timer->user_data` field when the timer event callback is called, attempted use of that pointer will likely cause a crash because the pointer is not valid any more. However, a timely dump of the Widget Tree right before that point will show that the Widget no longer exists.

### Find child by ID

#### Avvertimento

`lv_obj_find_by_id(widget, id)` is deprecated. To find a widget use `obj_name`.

`lv_obj_find_by_id(widget, id)` will perform a recursive walk through `widget`'s children and return the first child encountered having the given ID.

## 9.9 Widget Properties

Widget Properties provides a way to greatly reduce the size of the interface between LVGL and whatever logic layer is just above it, to get and set the most important properties of Widgets. It's intended use is to:

- simplify (decreasing development time) writing bindings for LVGL in another language, such as:
  - Micropython,
  - Lua,
  - Python,
  - Perl,
  - .NET
- make it possible to control the UI (or parts of it, e.g. animation) via external input, without modifying firmware, such as:
  - an external text file (YAML, JSON, XML, custom)
  - any external input source (e.g. serial)

While using it consumes more program space and more CPU overhead while setting and getting Widget properties, it is designed so minimize that additional CPU overhead.

### 9.9.1 What is a Widget Property?

A Widget's properties are the combined set of *Stili* plus additional properties that are unique to each type of Widget, that determine what the Widget looks like and how it behaves. Examples: size, position, color, are properties of all Widgets whereas text, long-mode, selection-start, and selection-end, are properties unique to Label Widgets. A Widget's *local styles* are also valid properties in this context.

The non-style Widget properties available for a given Widget are implemented at the top of that Widget's primary `.c` file as a `const` id-to-function-pointer lookup array, like this example for the Label Widget:

```
#if LV_USE_OBJ_PROPERTY
static const lv_property_ops_t properties[] = {
{
    .id = LV_PROPERTY_LABEL_TEXT,
    .setter = lv_label_set_text,
    .getter = lv_label_get_text,
},
}
```

(continues on next page)

(continua dalla pagina precedente)

```
{
    .id = LV_PROPERTY_LABEL_LONG_MODE,
    .setter = lv_label_set_long_mode,
    .getter = lv_label_get_long_mode,
},
{
    .id = LV_PROPERTY_LABEL_TEXT_SELECTION_START,
    .setter = lv_label_set_text_selection_start,
    .getter = lv_label_get_text_selection_start,
},
{
    .id = LV_PROPERTY_LABEL_TEXT_SELECTION_END,
    .setter = lv_label_set_text_selection_end,
    .getter = lv_label_get_text_selection_end,
},
};

#endif
```

This array is attached to the `properties` field of the Widget's class, so all Widgets of the same type share the same id-to-function-pointer lookup array.

Some properties are read-only. When this is the case, only the `getter` field in the corresponding array element will be initialized with a function pointer. Example: an object's child-Widget count or scroll position must be controlled via other types of input, but their values are readable through this API.

## 9.9.2 Utilizzo

By default, this feature of LVGL is turned off. It can be turned on by configuring `LV_USE_OBJ_PROPERTY` to 1 in `lv_conf.h`.

The 3 functions that then become available are:

- `lv_result_t lv_obj_set_property(widget, lv_property_t * value)` Sets specified property of Widget.
- `lv_property_t lv_obj_get_property(widget, lv_prop_id_t id)` Reads property value from Widget.
- `lv_result_t lv_obj_set_properties(widget, lv_property_t * values, count)` Sets multiple Widget properties from an array of `lv_property_t`.

An `lv_prop_id_t` is a *Property ID*, whereas an `lv_property_t` is a struct that pairs a *Property ID* with a *Property Value*.

The following is an example of an array that could be used as the `values` argument in `lv_obj_set_properties()`:

```
lv_property_t values[] = {
    { .id = LV_PROPERTY_IMAGE_SRC, .ptr = &img_demo_widgets_avatar, },
    { .id = LV_PROPERTY_IMAGE_PIVOT, .ptr = &pivot_50, },
    { .id = LV_PROPERTY_IMAGE_SCALE, .num = 128, },
    { .id = LV_PROPERTY_OBJ_FLAG_CLICKABLE, .num = 1, },
    { .id = LV_STYLE_IMAGE_OPA, .num = 128, },
    { .id = LV_STYLE_BG_COLOR, .color = (lv_color_t){.red = 0x11, .green = 0x22, .blue = 0x33}, },
}
```

Alternately, `lv_obj_set_property(widget, value)` could be called using this array's individual `value` elements inside a loop.

### Property ID

`lv_prop_id_t` identifies which property to get/set. It is an enum value defined in the primary `.h` file for the Widget in question. Because the actual names are "assembled" by a preprocessor string-concatenation macro and are thus hard to visualize, you can also find the names in the Widget's primary `.c` file in the `properties[]` array initializing the `.id` fields in the array. For example, `LV_PROPERTY_LABEL_TEXT` is one found in `lv_label.c`, and the properties available to all Widgets are found near the top of the `lv_obj.c` file.

That array is attached to the Widget's class, enabling "getter" and "setter" functions to be looked up for each type of Widget where Widget properties has been implemented. (Note: this is done internally so you don't have to.)

If the property you need to set or get using this API is not implemented yet, you can add your own Widget property ID following same rules and using one of two helper macros in the `enum` in the Widget's primary `.h` file. In both cases, the "assembled" value is a 32-bit value:

- **LV\_PROPERTY\_ID** (for single values -- see *Single Values* below); bits <31:28> contain the property's value type and bits <27:0> contain the property ID.
- **LV\_PROPERTY\_ID2** (for paired values -- see *Paired Values* below); bits <31:28> contain the type for the property's 1st value, bits <27:24> contain the type for the 2nd value, and bits <23:0> contain the property ID.

Just make sure the ID is unique across all Widgets.

Note that `lv_style_prop_t` (enumerator values beginning with `LV_PROPERTY_STYLE_...`) are also valid property IDs, and can be used to set or get a Widget's style values.

## Property Value

`lv_property_t` is a struct that begins with an `id` field whose meaning is the same as property ID described above, paired with a value, which is a union of all possible property types including integer, pointer and color. The `value` field is also capable of carrying the different value types for styles. It does this by being a union of all the different types that might be needed. The list of "union-ed" fields at this writing are:

## Single Values

```
int32_t      num;      /*< Signed integer number (enums or "normal" numbers) */
uint32_t     num_u;    /*< Unsigned integer number (opacity, Booleans) */
bool         enable;   /*< Booleans */
const void*  *ptr;     /*< Constant pointers (font, cone text, etc.) */
lv_color_t   color;    /*< Colors */
lv_value_precise_t precise; /*< float or int for precise value */
lv_point_t   point;   /*< Point, contains two int32_t */

struct {
    /*
     * Note that place struct member `style` at first place is intended.
     * `style` shares same memory with `num`, `ptr`, `color`.
     * So we set the style value directly without using `prop.style.num`.
     *
     * E.g.
     *
     * static const lv_property_t obj_pos_x = {
     *     .id = LV_PROPERTY_STYLE_X,
     *     .num = 123,
     *     .selector = LV_STATE_PRESSED,
     * };
     *
     * instead of:
     * static const lv_property_t obj_pos_x = {
     *     .id = LV_PROPERTY_STYLE_X,
     *     .style.num = 123, // note this line.
     *     .selector = LV_STATE_PRESSED,
     * };
     */
    lv_style_value_t style;  /*< Make sure it's the first element in struct. */
    uint32_t        selector; /*< Style selector, lv_part_t | lv_state_t */
};
```

## Paired Values

```
/*
 * For some properties like slider range, it contains two simple (4-byte) values
 * so we can use 'arg1.num' and 'arg2.num' to set the argument.
 */
struct {
    union {
        int32_t      num;
        uint32_t     num_u;
        bool         enable;
        const void*  *ptr;
        lv_color_t   color;
        lv_value_precise_t precise;
    } arg1, arg2;
};
```

You can find the current `lv_property_t` struct in the `lv_obj_property.h` file.

## Property ID Lookup by Name

Setting configuration macro `LV_USE_OBJ_PROPERTY_NAME` to 1 enables the following functions to look up property IDs by passing property name (a string):

- `lv_prop_id_t lv_obj_property_get_id(widget, name)` Gets property ID by recursively searching for `name` in Widget's class hierarchy, and if still not found, then searches style properties.
- `lv_prop_id_t lv_obj_class_property_get_id(class_p, name)` Gets property ID by doing a non-recursive search for `name` directly in Widget class properties.

- `lv_prop_id_t lv_style_property_get_id(name)` Gets style property ID by name.

The latter two functions are useful when you already know `name` is among the properties of a specific Widget class, or is a style name, since a property name may exist in both lists. Because of the search sequence in `lv_obj_property_get_id(widget, name)`, if a name does exist in both lists, then using this function forces the name in the Widget's class hierarchy properties to have precedence over the style name.

You can tell which names are available by looking in the `.c` files in the `./src/widgets/property/` directory. Note that to support binary name searches, these arrays are generated so that they are guaranteed to be in alphabetical order. If you need to add a property that is not present, it is recommended to add it in the `enum` near the top of the Widget's primary `.h` file, and re-generate these lists using `./scripts/properties.py` to ensure alphabetical ordering is preserved.

### 9.9.3 Additional Notes

For the `lv_property_t * value` argument of the `lv_obj_set_property()` function, the language used to call that function (e.g. in a static or dynamically-loaded library) may need additional code to convert values from their local data type (e.g. dict, table, etc.) to a C struct before passing it to the `lv_obj_set_property()` function.

### 9.9.4 API

`lv_obj_property_names.h`

## 9.10 Observer

### 9.10.1 Come si Usa

#### Panoramica

The `lv_observer` module is an implementation of the [Observer Pattern](#).

This implementation consists of:

#### Subjects

(in global memory or heap) are "logic packages", each containing the value being "observed" and its type (integer (`int32_t`), a string, a pointer, an `lv_color_t`, a `float`, or a group);

#### Observers

(zero or more per Subject, always dynamically-allocated) are always attached to exactly one Subject, and provide user-defined notifications each the time Subject's value changes.

A Subject and its Observers can be used in various ways:

1. Simply subscribe to a Subject and get notified when the Subject's value changes.
2. Subscribe to a group Subject (connects a group of Subjects) to get notified when any of the Subjects' values change in the group.
3. Bind Widgets to Subjects to automatically match the Widget's value with the Subject (e.g. a Label's text or an Arc's value).

#### Utilizzo

Using Observer first requires `LV_USE_OBSERVER` be configured to 1. (It is 1 by default, and can be set to 0 to save some program space if you will not be using Observer.)

A typical use case looks like this:

```
// Any typical global variable
lv_subject_t my_subject;

/*-----
 * main.c
 *-----*/
```

(continues on next page)

(continua dalla pagina precedente)

```

extern lv_subject_t my_subject;

void main(void)
{
    // Initialize Subject as integer with the default value of 10.
    lv_subject_init_int(&my_subject, 10);

    some_module_init();
}

/*-----
 * some_module.c
 *-----*/
extern lv_subject_t my_subject;

// Will be called when Subject's value changes
static void some_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    int32_t v = lv_subject_get_int(subject);
    do_something(v);
}

void some_module_init(void)
{
    // Subscribe to Subject as an Observer.
    lv_subject_add_observer(&my_subject, some_observer_cb, NULL);
}

/*-----
 * some_system.c
 *-----*/
extern lv_subject_t my_subject;

void some_event(void)
{
    // The below call sets Subject's value to 30 and notifies current Observers.
    lv_subject_set_int(&my_subject, 30);
}

```

## Subject

### Subject Initialization

Subjects have to be static or global variables, or dynamically-allocated `lv_subject_t` objects. Reason: their content must remain valid through the life of the Subject.

To initialize a Subject use `lv_subject_init_<type>(&subject, params, init_value)`. The following initialization functions exist, one for each of the Subject types:

#### **Integer**

void `lv_subject_init_int(subject, int_value)`

#### **Float**

void `lv_subject_init_float(subject, float_value)`

#### **String**

void `lv_subject_init_string(subject, buf, prev_buf, buf_size, initial_string)`

#### **Pointer**

void `lv_subject_init_pointer(subject, ptr)`

#### **Color**

void `lv_subject_init_color(subject, color)`

#### **Group**

void `lv_subject_init_group(group_subject, subject_list[], count)`

### Setting a Subject's Value

The following functions are used to update a Subject's value:

#### **Integer**

void `lv_subject_set_int(subject, int_value)`

#### **String**

void `lv_subject_copy_string(subject, buf)`

#### **Pointer**

void `lv_subject_set_pointer(subject, ptr)`

**Color**

```
void lv_subject_set_color(subject, color)
```

At the end of each of these calls, if the new value differs from the previous value, a notification is sent to all current Observers.

**Getting a Subject's Value**

The following functions are used to get a Subject's current value:

**Integer**

```
int32_t lv_subject_get_int(subject)
```

**Float**

```
float lv_subject_get_float(subject)
```

**String**

```
const char * lv_subject_get_string(subject)
```

**Pointer**

```
const void * lv_subject_get_pointer(subject)
```

**Color**

```
lv_color_t lv_subject_get_color(subject)
```

**Getting a Subject's Previous Value**

The following functions are used to get a Subject's previous value:

**Integer**

```
int32_t lv_subject_get_previous_int(subject)
```

**Float**

```
float lv_subject_get_previous_float(subject)
```

**String**

```
const char * lv_subject_get_previous_string(subject)
```

**Pointer**

```
const void * lv_subject_get_previous_pointer(subject)
```

**Color**

```
lv_color_t lv_subject_get_previous_color(subject)
```

**Observer****Subscribing to a Subject**

The action of subscribing to a Subject:

- dynamically allocates an Observer object,
- attaches it to the Subject,
- performs an initial notification to the Observer (allowing the Observer to update itself with the Subject's current value), and
- returns a pointer to the newly-created Observer.

Thereafter the Observer will receive a notification each time the Subject's value changes, as long as that Observer remains attached (subscribed) to that Subject.

Notifications are performed by calling the callback function provided when subscribing to the Subject.

## Simple Subscription

To subscribe to a Subject one of the `lv_subject_add_observer...` functions are used. This is covered below.

Alternately, if you want to bind a Subject's value to a Widget's property, one of the `lv_<widget_type>_bind...` functions can be used. See [Widget Binding](#) for more details.

By using `lv_obj_add_subject...` it's also possible to change a subject's value on a trigger. It's covered in [Change Subject on Event](#)

For the most basic use case, subscribe to a Subject by using the following function:

```
lv_observer_t * observer = lv_subject_add_observer(&some_subject,
    some_observer_cb, user_data)
```

where the Observer's notification callback should look like this:

```
static void some_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    ...
}
```

This function returns a pointer to the newly-created Observer.

When using this method of subscribing, it is the responsibility of the user to call `lv_observer_remove(observer)` when the Observer is no longer needed, which both unsubscribes it from the Subject and deletes it from the LVGL heap.

## Subscribing While Associating Observer with a Non-Widget Object

The function subscribes to a Subject additionally associates the Observer with a pointer to any type of object, a copy of which is saved in the Observer's `target` field. This function should be used when the pointer *does not* point to a Widget.

```
lv_observer_t * observer = lv_subject_add_observer_with_target(&some_subject, some_observer_cb, some_pointer,
    user_data)
```

A copy of the passed pointer can be retrieved by calling `lv_observer_get_target(observer)`, e.g. inside the callback function.

When using this method of subscribing, it is the responsibility of the user to call `lv_observer_remove(observer)` when the Observer is no longer needed, which both unsubscribes it from the Subject and deletes it from the LVGL heap.

## Subscribing While Associating Observer with a Widget

The function below associates a Widget with an Observer while subscribing to a Subject. A copy of the pointer to that Widget is saved in the Observer's `target` field. This works exactly like the above method except that when the Widget is deleted, the Observer thus created will be automatically unsubscribed from the Subject and deleted from the LVGL heap. Note this is different from [Widget Binding](#).

```
lv_observer_t * observer = lv_subject_add_observer_obj(&some_subject,
    some_observer_cb, widget, user_data)
```

Any number of Observers can be created and be associated with a Widget this way.

A copy of the pointer to the Widget can be retrieved by calling `lv_observer_get_target_obj(observer)`, e.g. inside the callback function. Note that this function returns the stored pointer as a `lv_obj_t *` type, as opposed to the `void *` type returned by `lv_observer_get_target(observer)`. (`lv_observer_get_target(observer)` can still be used if you need that pointer as a `void *` type for any reason, but in practice, this would be rare.)

## Unsubscribing from a Subject

When an Observer and its associated events are no longer needed, you can remove it from memory using any of the methods below, depending on the effect you want to create:

- If Widget needs to be deleted, simply delete the Widget, which will automatically gracefully remove the Observer (and its events) from the Subject, while deleting the Widget.
- If Widget does NOT need to be deleted:
  - `lv_obj_remove_from_subject(widget, subject)` deletes all Observers associated with `widget`. `subject` can be NULL to unsubscribe the Widget from all associated Subjects.
  - `lv_subject_deinit(subject)` gracefully disconnects `subject` from all associated Observers and Widget events. This includes subscriptions made using any of the *Widget Binding* functions covered below.
  - `lv_observer_remove(observer)` deletes that specific Observer and gracefully disconnects it from its `subject` and any associated Widgets, where `observer` is the return value any of the above `lv_subject_add_observer_...()` functions.

## Subject Groups

When something in your system relies on more than one value (i.e. it needs to be notified when any of a SET of two or more values changes), it can be made an Observer of a Subject Group.

Let us consider an example of an instrument which measures either voltage or current. To display the measured value on a label, 3 things are required:

1. What is being measured (current or voltage)?
2. What is the measured value?
3. What is the range or unit ("mV", "V", "mA", "A")?

When any of these 3 input values change, the label needs to be updated, and it needs to know all 3 values to compose its text.

To handle this you can create an array from the addresses of all the Subjects that are relied upon, and pass that array as a parameter when you initialize a Subject with GROUP type.

```
static lv_subject_t * subject_list[3] = {&subject_1, &subject_2, &subject_3};
lv_subject_init_group(&subject_all, subject_list, 3); /* Last argument is number of elements. */
```

Observers are then added to Subject Groups (e.g. `subject_all`) in the usual way. When this is done, a change to the value of any of the Subjects in the group triggers a notification to all Observers subscribed to the Subject Group (e.g. `subject_all`).

As an example, the above scenario with Voltage/Current measurement might look like this:

```
lv_obj_t * label = lv_label_create(lv_screen_active());

lv_subject_t subject_mode; // Voltage or Current
lv_subject_t subject_value; // Measured value
lv_subject_t subject_unit; // The unit
lv_subject_t subject_all; // Subject group that connects the above 3 Subjects
lv_subject_t * subject_list[3] = {&subject_mode, &subject_value, &subject_unit}; // The elements of the group

lv_subject_init_int(&subject_mode, 0); // Let's say 0 is Voltage, 1 is Current
lv_subject_init_int(&subject_value, 0);
lv_subject_init_pointer(&subject_unit, "V");
lv_subject_init_group(&subject_all, subject_list, 3);

lv_subject_add_observer_obj(&subject_all, all_observer_cb, label, NULL);
...

static void all_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_obj_t * label = lv_observer_get_target(observer);
    lv_subject_t * subject_mode = lv_subject_get_group_element(subject, 0);
    lv_subject_t * subject_value = lv_subject_get_group_element(subject, 1);
    lv_subject_t * subject_unit = lv_subject_get_group_element(subject, 2);

    int32_t mode = lv_subject_get_int(subject_mode);
    int32_t value = lv_subject_get_int(subject_value);
    const char * unit = lv_subject_get_pointer(subject_unit);

    lv_label_set_text_fmt(label, "%s: %d %s", mode ? "Current" : "Voltage", value, unit);
}
```

## Widget Binding

The following methods of subscribing to an integer-type Subject associate the Observer with ONE of a Widget's properties as thought that property itself were the Observer. Any of the following Widget properties can be thus bound to an Subject's integer value:

- flag (or OR-ed combination of flags) from the `LV_OBJ_FLAG_...` enumeration values;
- state (or OR-ed combination of states) from the `LV_STATE_...` enumeration values;
- text value for Label Widgets;
- integer value for these Widget types:
  - Arco
  - Drop-Down
  - Roller
  - Slider

Any number of Observers can be created for a single Widget, each bound to ONE of the above properties.

For all of the `lv_..._bind_...` functions covered below, they are similar to `lv_subject_add_observer_obj(&some_subject, some_observer_cb, widget, user_data)` in that they create an Observer and associates the Widget with it. What is different is that updates to the Widget's property thus bound are handled internally -- the user *does not supply callback functions* for any of these subscribing methods -- the callback methods are supplied by the Observer subsystem.

### **i** Nota

While the examples below show saving a reference to the created Observer objects for the various `lv_..._bind_...` functions, it is not necessary to do so unless you need them for some purpose, because the created Observer objects will be automatically deleted when the Widget is deleted.

## Any Widget Type

### I Flag

The following methods of subscribing to an integer Subject affect a Widget's flag (or OR-ed combination of flags). When the subscribing occurs, and each time the Subject's value is changed thereafter, the Subject's value is compared with the specified reference value, and the specified flag(s) is (are):

- SET when the Subject's integer value fulfills the indicated condition, and
- CLEARED otherwise.

Here are the functions that carry out this method of subscribing to a Subject. The `flags` argument can contain a single, or bit-wise OR-ed combination of any of the `LV_OBJ_FLAG_...` enumeration values.

```

equal
  lv_obj_bind_flag_if_eq(widget, &subject, flags, ref_value)

not equal
  lv_obj_bind_flag_if_not_eq(widget, &subject, flags, ref_value)

greater than
  lv_obj_bind_flag_if_gt(widget, &subject, flags, ref_value)

greater than or equal
  lv_obj_bind_flag_if_ge(widget, &subject, flags, ref_value)

less than
  lv_obj_bind_flag_if_lt(widget, &subject, flags, ref_value)

```

**less than or equal**

```
lv_obj_bind_flag_if_le(widget, &subject, flags, ref_value)
```

**States**

The following methods of subscribing to an integer Subject affect a Widget's states (or OR-ed combination of states). When the subscribing occurs, and each time the Subject's value is changed thereafter, the Subject's value is compared with the specified reference value, and the specified state(s) is (are):

- SET when the Subject's integer value fulfills the indicated condition, and
- CLEARED otherwise.

Here are the functions that carry out this method of subscribing to a Subject. The **states** argument can contain a single, or bit-wise OR-ed combination of any of the **LV\_STATE\_...** enumeration values.

**equal**

```
lv_obj_bind_state_if_eq(widget, &subject, states, ref_value)
```

**not equal**

```
lv_obj_bind_state_if_not_eq(widget, &subject, states,
                           ref_value)
```

**greater than**

```
lv_obj_bind_state_if_gt(widget, &subject, states, ref_value)
```

**greater than or equal**

```
lv_obj_bind_state_if_ge(widget, &subject, states, ref_value)
```

**less than**

```
lv_obj_bind_state_if_lt(widget, &subject, states, ref_value)
```

**less than or equal**

```
lv_obj_bind_state_if_le(widget, &subject, states, ref_value)
```

**Checked State**

The following method of subscribing to an integer Subject affects a Widget's **LV\_STATE\_CHECKED** state. When the subscribing occurs, and each time the Subject's value is changed thereafter, the Subject's value is compared to a reference value of **0**, and the **LV\_STATE\_CHECKED** state is:

- CLEARED when the Subject's value is **0**, and
- SET when the Subject's integer value is non-zero.

Note that this is a two-way binding (Subject <====> Widget) so direct (or programmatic) interaction with the Widget that causes its **LV\_STATE\_CHECKED** state to be SET or CLEARED also causes the Subject's value to be set to **1** or **0** respectively.

- `lv_obj_bind_checked(widget, &subject)`

**Label Widgets**

This method of subscribing to an integer Subject affects a Label Widget's **text**. The Subject can be an STRING, POINTER or INTEGER type.

When the subscribing occurs, and each time the Subject's value is changed thereafter, the Subject's value is used to update the Label's text as follows:

**string Subject**

Subject's string is used to directly update the Label's text.

**pointer Subject**

If **NULL** is passed as the **format\_string** argument when subscribing, the Subject's pointer value is assumed to point to a NUL-terminated string. and is used to directly update the Label's text. See [The format\\_string Argument](#) for other options.

**integer Subject**

Subject's integer value is used with the `format_string` argument. See [The format\\_string Argument](#) for details.

Note that this is a one-way binding (Subject ==> Widget).

- `lv_label_bind_text(label, &subject, format_string)`

**The format\_string Argument**

The `format_string` argument is optional and if provided, must contain exactly 1 printf-like format specifier and be one of the following:

**string or pointer Subject**

"%s" to format the new pointer value as a string or "%p" to format the pointer as a pointer (typically the pointer's address value is spelled out with 4, 8 or 16 hexadecimal characters depending on the platform).

**integer Subject**

"%d" format specifier ("%d PRIidx --- a cross-platform equivalent where xx can be 8, 16, 32 or 64, depending on the platform).

**float Subject**

"%f" format specifier, or e.g. "%0.2f" to display two digits after the decimal point.

If NULL is passed for the `format_string` argument:

**string or pointer Subject**

Updates expect the pointer to point to a NUL-terminated string.

**integer Subject**

The Label will simply display the number. Equivalent to "%d".

**float Subject**

The Label will display the value with "%0.1f" format string.

**Example:** "%d °C"

**Arc Widgets**

This method of subscribing to an integer Subject affects an Arc Widget's integer value directly. Note that this is a two-way binding (Subject <==> Widget) so an end user's direct interaction with the Arc Widget updates the Subject's value and vice versa. (Requires `LV_USE_ARC` to be configured to 1.)

It support integer and float subjects.

- `lv_arc_bind_value(arc, &subject)`

**Slider Widgets**

This method of subscribing to an integer Subject affects a Slider Widget's integer value directly. Note that this is a two-way binding (Subject <==> Widget) so an end user's direct interaction with the Slider Widget updates the Subject's value and vice versa. (Requires `LV_USE_SLIDER` to be configured to 1.)

It support integer and float subjects.

- `lv_slider_bind_value(slider, &subject)`

**Roller Widgets**

This method of subscribing to an integer Subject affects a Roller Widget's integer value directly. Note that this is a two-way binding (Subject <==> Widget) so an end user's direct interaction with the Roller Widget updates the Subject's value and vice versa. (Requires `LV_USE_ROLLER` to be configured to 1.)

It support only integer subjects.

- `lv_roller_bind_value(roller, &subject)`

## Drop-Down Widgets

This method of subscribing to an integer Subject affects a Drop-Down Widget's integer value directly. Note that this is a two-way binding (Subject <==> Widget) so an end user's direct interaction with the Drop-Down Widget updates the Subject's value and vice versa. (Requires `LV_USE_DROPDOWN` to be configured to 1.)

It support only integer subjects.

- `lv_dropdown_bind_value(dropdown, &subject)`

## Change Subject on Event

It's a common requirement to update a subject based on a user action (trigger). To simplify this, *subject set* and *increment* actions can be attached directly to any widget.

Internally, these are implemented as special event callbacks. Note: these callbacks are **not** automatically removed when a subject is deinitiated.

### Increment

`lv_obj_add_subject_increment_event(obj, subject, step, min, max)` Increments the subject's value by `step`, clamped between `min` and `max`.

Ad esempio:

```
lv_obj_add_subject_increment_event(button1, subject1, LV_EVENT_CLICKED, 5, -10, 80)
```

This will increment `subject1` by 5 when `button1` is clicked. The resulting value will be constrained to the range -10 to 80.

Using a negative `step` will decrement the value instead.

### Set to a Value

- `lv_obj_add_subject_set_int_event(obj, subject, trigger, value)`
- `lv_obj_add_subject_set_string_event(obj, subject, trigger, text)`

These functions set the given subject (integer or string) to a fixed value when the specified trigger event occurs.

## API

`lv_observer.h`

`lv_observer_private.h`

## 9.10.2 Esempi

### Bind a slider's value to a label

```
#include "../../lv_examples.h"
#if LV_USE_OBSERVER && LV_USE_SLIDER && LV_USE_LABEL && LV_BUILD_EXAMPLES

static lv_subject_t temperature_subject;

/***
 * A slider sends a message on value change and a label display's that value
 */
void lv_example_observer_1(void)
{
    lv_subject_init_int(&temperature_subject, 28);

    /*Create a slider in the center of the display*/
    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_slider_bind_value(slider, &temperature_subject);

    /*Create a label below the slider*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
```

(continues on next page)

(continua dalla pagina precedente)

```

    lv_obj_align(label, LV_ALIGN_CENTER, 0, 30);
    lv_label_bind_text(label, &temperature_subject, "%d °C");
}

#endif

```

## Handling login and its states

```

#include "../../lv_examples.h"
#if LV_USE_OBSERVER && LV_USE_SLIDER && LV_USE_LABEL && LV_BUILD_EXAMPLES

/*This is the only interface between the UI and the application*/
static lv_subject_t engine_subject;

static void app_init(void);
static void ui_init(void);

/***
 * Simple PIN login screen to start an engine.
 * The only interface between the UI and the application is a single "subject".
 */
void lv_example_observer_2(void)
{
    lv_subject_init_int(&engine_subject, 0);
    app_init();
    ui_init();
}

/*
 * APPLICATION
 *
 * This part contains a demo application logic.
 * It doesn't know anything about the internals of the UI
 * and uses any the 'engine_subject' as an interface.
 */
static void engine_state_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    LV_UNUSED(observer);

    int32_t v = lv_subject_get_int(subject);
    LV_UNUSED(v);
    /*In a real application set/clear a pin here*/
    LV_LOG_USER("Engine state: %" LV_PRIId32, v);
}

static void app_init(void)
{
    lv_subject_add_observer(&engine_subject, engine_state_observer_cb, NULL);
}

/*
 * USER INTERFACE
 *
 * This part contains only UI related code and data.
 * In a project it would a separate file and the
 * application couldn't see its internals
 */
typedef enum {
    LOGGED_OUT,
    LOGGED_IN,
    AUTH_FAILED,
} auth_state_t;

static lv_subject_t auth_state_subject;

static void textarea_event_cb(lv_event_t * e)
{
    lv_obj_t * ta = lv_event_get_target_obj(e);
    if(lv_strcmp(lv_textarea_get_text(ta), "hello") == 0) {
        lv_subject_set_int(&auth_state_subject, LOGGED_IN);
    }
    else {
        lv_subject_set_int(&auth_state_subject, AUTH_FAILED);
    }
}

static void info_label_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_obj_t * label = (lv_obj_t *) lv_observer_get_target(observer);
    switch(lv_subject_get_int(subject)) {
        case LOGGED_IN:
            lv_label_set_text(label, "Login successful");
            break;
        case LOGGED_OUT:
            lv_label_set_text(label, "Logged out");
            break;
        case AUTH_FAILED:
            lv_label_set_text(label, "Login failed");
            break;
    }
}

static void log_out_click_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);
    lv_subject_set_int(&auth_state_subject, LOGGED_OUT);
}

static void ui_init(void)
{
    lv_subject_init_int(&auth_state_subject, LOGGED_OUT);

    /*Create a slider in the center of the display*/
    lv_obj_t * ta = lv_textarea_create(lv_screen_active());
    lv_obj_set_pos(ta, 10, 10);
    lv_obj_set_width(ta, 200);
    lv_textarea_set_one_line(ta, true);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_textarea_set_password_mode(ta, true);
lv_textarea_set_placeholder_text(ta, "The password is: hello");
lv_obj_add_event_cb(ta, textarea_event_cb, LV_EVENT_READY, NULL);
lv_obj_bind_state_if_eq(ta, &auth_state_subject, LV_STATE_DISABLED, LOGGED_IN);

lv_obj_t * kb = lv_keyboard_create(lv_screen_active());
lv_keyboard_set_textarea(kb, ta);

lv_obj_t * btn;
lv_obj_t * label;

/*Create a log out button which will be active only when logged in*/
btn = lv_button_create(lv_screen_active());
lv_obj_set_pos(btn, 220, 10);
lv_obj_add_event_cb(btn, log_out_click_event_cb, LV_EVENT_CLICKED, NULL);
lv_obj_bind_state_if_not_eq(btn, &auth_state_subject, LV_STATE_DISABLED, LOGGED_IN);

label = lv_label_create(btn);
lv_label_set_text(label, "LOG OUT");

/*Create a label to show info*/
label = lv_label_create(lv_screen_active());
lv_obj_set_pos(label, 10, 60);
lv_subject_add_observer_obj(&auth_state_subject, info_label_observer_cb, label, NULL);

/*Create button which will be active only when logged in*/
btn = lv_button_create(lv_screen_active());
lv_obj_set_pos(btn, 10, 80);
lv_obj_add_flag(btn, LV_OBJ_FLAG_CHECKABLE);
lv_obj_bind_state_if_not_eq(btn, &auth_state_subject, LV_STATE_DISABLED, LOGGED_IN);
lv_button_bind_checked(btn, &engine_subject);
label = lv_label_create(btn);
lv_label_set_text(label, "START ENGINE");
}

#endif

```

## Set time with 12/24 mode and AM/PM

```

#include "../../lv_examples.h"
#if LV_USE_OBSERVER && LV_USE_SLIDER && LV_USE_LABEL && LV_USE_ROLLER && LV_USE_DROPDOWN && LV_FONT_MONTserrat_30 && LV_BUILD_EXAMPLES

static lv_subject_t hour_subject;
static lv_subject_t minute_subject;
static lv_subject_t format_subject;
static lv_subject_t am_pm_subject;
static lv_subject_t time_subject;
static lv_subject_t * time_group_array_subject[] = {&hour_subject, &minute_subject, &format_subject, &am_pm_subject};
const char * hour12_options = "01\n02\n03\n04\n05\n06\n07\n08\n09\n10\n11\n12\n13\n14\n15\n16\n17\n18\n19\n20\n21\n22\n23";
const char * hour24_options =
"00\n01\n02\n03\n04\n05\n06\n07\n08\n09\n10\n11\n12\n13\n14\n15\n16\n17\n18\n19\n20\n21\n22\n23\n24\n25\n26\n27\n28\n29\n30\n31\n32\n33\n34\n35";
const char * minute_options =
"00\n01\n02\n03\n04\n05\n06\n07\n08\n09\n10\n11\n12\n13\n14\n15\n16\n17\n18\n19\n20\n21\n22\n23\n24\n25\n26\n27\n28\n29\n30\n31\n32\n33\n34\n35\n36\n37\n38\n39\n40\n41\n42\n43\n44\n45\n46\n47\n48\n49\n50\n51\n52\n53\n54\n55\n56\n57\n58\n59";

static void set_btn_clicked_event_cb(lv_event_t * e);
static void close_clicked_event_cb(lv_event_t * e);
static void hour_roller_options_update(lv_observer_t * observer, lv_subject_t * subject);
static void time_observer_cb(lv_observer_t * observer, lv_subject_t * subject);

typedef enum {
    TIME_FORMAT_12,
    TIME_FORMAT_24,
} time_format_t;

typedef enum {
    TIME_AM,
    TIME_PM,
} time_am_pm_t;

/***
 * Show how to handle a complex time setting with hour, minute, 12/24 hour mode, and AM/PM switch
 * In a real application the time can be displayed on multiple screens and it's not trivial
 * how and where to store the current values and how to get them.
 * In this example the widgets to set the time are created/deleted dynamically,
 * yet they always know what the current values are by using subjects.
 */
void lv_example_observer_3(void)
{
    /*Initialize the subjects.
     *The UI will update these and read the current values from here,
     *however the application can update these values at any time and
     *the widgets will be updated automatically.*/
    lv_subject_init_int(&hour_subject, 7);
    lv_subject_init_int(&minute_subject, 45);
    lv_subject_init_int(&format_subject, TIME_FORMAT_12);
    lv_subject_init_int(&am_pm_subject, TIME_AM);
    lv_subject_init_group(&time_subject, time_group_array_subject, 4);

    /*Create the UI*/
    lv_obj_t * time_label = lv_label_create(lv_screen_active());
    lv_obj_set_style_text_font(time_label, &lv_font_montserrat_30, 0);
    lv_subject_add_observer_obj(&time_subject, time_observer_cb, time_label, NULL);
    lv_obj_set_pos(time_label, 24, 24);

    lv_obj_t * set_btn = lv_button_create(lv_screen_active());
    lv_obj_set_pos(set_btn, 180, 24);
    lv_obj_add_event_cb(set_btn, set_btn_clicked_event_cb, LV_EVENT_CLICKED, NULL);

    lv_obj_t * set_label = lv_label_create(set_btn);
    lv_label_set_text(set_label, "Set");

    /*Update some subjects to see if the UI is updated as well*/
    lv_subject_set_int(&hour_subject, 9);
    lv_subject_set_int(&minute_subject, 30);
    lv_subject_set_int(&am_pm_subject, TIME_PM);
}

```

(continues on next page)

(continua dalla pagina precedente)

```

static void set_btn_clicked_cb(lv_event_t * e)
{
    lv_obj_t * set_btn = lv_event_get_target_obj(e);
    lv_obj_add_state(set_btn, LV_STATE_DISABLED);

    lv_obj_t * cont = lv_obj_create(lv_screen_active());
    lv_obj_set_size(cont, lv_pct(100), LV_SIZE_CONTENT);
    lv_obj_align(cont, LV_ALIGN_BOTTOM_MID, 0, 0);

    lv_obj_t * hour_roller = lv_roller_create(cont);
    lv_obj_add_flag(hour_roller, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK);
    lv_subject_add_observer_obj(&format_subject, hour_roller_options_update, hour_roller, NULL);
    lv_roller_bind_value(hour_roller, &hour_subject);
    lv_obj_set_pos(hour_roller, 0, 0);

    lv_obj_t * min_roller = lv_roller_create(cont);
    lv_roller_set_options(min_roller, minute_options, LV_ROLLER_MODE_NORMAL);
    lv_roller_bind_value(min_roller, &minute_subject);
    lv_obj_set_pos(min_roller, 64, 0);

    lv_obj_t * format_dropdown = lv_dropdown_create(cont);
    lv_dropdown_set_options(format_dropdown, "12\n24");
    lv_dropdown_bind_value(format_dropdown, &format_subject);
    lv_obj_set_pos(format_dropdown, 128, 0);
    lv_obj_set_width(format_dropdown, 80);

    lv_obj_t * am_pm_dropdown = lv_dropdown_create(cont);
    lv_dropdown_set_options(am_pm_dropdown, "am\npm");
    lv_dropdown_bind_value(am_pm_dropdown, &am_pm_subject);
    lv_obj_bind_state_if_eq(am_pm_dropdown, &format_subject, LV_STATE_DISABLED, TIME_FORMAT_24);
    lv_obj_set_pos(am_pm_dropdown, 128, 48);
    lv_obj_set_width(am_pm_dropdown, 80);

    lv_obj_t * close_btn = lv_button_create(cont);
    lv_obj_align(close_btn, LV_ALIGN_TOP_RIGHT, 0, 0);
    /*Pass the set_btn as user_data to make it non-disabled on close*/
    lv_obj_add_event_cb(close_btn, close_clicked_event_cb, LV_EVENT_CLICKED, set_btn);

    lv_obj_t * close_label = lv_label_create(close_btn);
    lv_label_set_text(close_label, LV_SYMBOL_CLOSE);
}

static void close_clicked_event_cb(lv_event_t * e)
{
    lv_obj_t * set_btn = (lv_obj_t *) lv_event_get_user_data(e);
    lv_obj_t * close_btn = lv_event_get_target_obj(e);
    lv_obj_t * cont = lv_obj_get_parent(close_btn);
    lv_obj_remove_state(set_btn, LV_STATE_DISABLED);
    lv_obj_delete(cont);
}

/*Watch all related subject to display the current time correctly*/
static void time_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    int32_t hour = lv_subject_get_int(lv_subject_get_group_element(subject, 0));
    int32_t minute = lv_subject_get_int(lv_subject_get_group_element(subject, 1));
    int32_t format = lv_subject_get_int(lv_subject_get_group_element(subject, 2));
    int32_t am_pm = lv_subject_get_int(lv_subject_get_group_element(subject, 3));

    lv_obj_t * label = (lv_obj_t *) lv_observer_get_target(observer);

    if(format == TIME_FORMAT_24) {
        lv_label_set_text_fmt(label, "%" LV_PRId32 ":" LV_PRId32, hour, minute);
    }
    else {
        lv_label_set_text_fmt(label, "%" LV_PRId32 ":" LV_PRId32 "%s", hour + 1, minute, am_pm == TIME_AM ? "am" : "pm");
    }
}

/*Change the hour options on format change*/
static void hour_roller_options_update(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_obj_t * roller = (lv_obj_t *) lv_observer_get_target(observer);
    int32_t prev_selected = lv_roller_get_selected(roller);
    int32_t v = lv_subject_get_int(subject);
    if(v == TIME_FORMAT_12) {
        prev_selected--;
        if(prev_selected > 12) prev_selected -= 12;
        lv_roller_set_options(roller, hour12_options, LV_ROLLER_MODE_NORMAL);
    }
    else {
        prev_selected++;
        lv_roller_set_options(roller, hour24_options, LV_ROLLER_MODE_NORMAL);
    }

    lv_roller_set_selected(roller, prev_selected, LV_ANIM_OFF);
    lv_obj_send_event(roller, LV_EVENT_VALUE_CHANGED, NULL);
}

#endif

```

## Custom tab view with state management

```

#include "../../lv_examples.h"
#if LV_USE_OBSERVER && LV_USE_SLIDER && LV_USE_LABEL && LV_USE_ROLLER && LV_USE_DROPDOWN && LV_FONT_MONTserrat_30 && LV_BUILD_EXAMPLES

static void cont_observer_cb(lv_observer_t * observer, lv_subject_t * subject);
static void btn_create(lv_obj_t * parent, const char * text);
static void btn_click_event_cb(lv_event_t * e);
static void btn_observer_cb(lv_observer_t * observer, lv_subject_t * subject);
static void indicator_observer_cb(lv_observer_t * observer, lv_subject_t * subject);

static lv_subject_t current_tab_subject;
static lv_subject_t slider_subject[4];
static lv_subject_t dropdown_subject[3];
static lv_subject_t roller_subject[2];

void lv_example_observer_4(void)

```

(continues on next page)

(continua dalla pagina precedente)

```
{
    lv_subject_init_int(&current_tab_subject, 0);
    lv_subject_init_int(&slider_subject[0], 0);
    lv_subject_init_int(&slider_subject[1], 0);
    lv_subject_init_int(&slider_subject[2], 0);
    lv_subject_init_int(&slider_subject[3], 0);
    lv_subject_init_int(&dropdown_subject[0], 0);
    lv_subject_init_int(&dropdown_subject[1], 0);
    lv_subject_init_int(&dropdown_subject[2], 0);
    lv_subject_init_int(&roller_subject[0], 0);
    lv_subject_init_int(&roller_subject[1], 0);

    lv_obj_t * main_cont = lv_obj_create(lv_screen_active());
    lv_obj_remove_style_all(main_cont);
    lv_obj_set_size(main_cont, lv_pct(100), lv_pct(100));
    lv_obj_set_style_pad_all(main_cont, 0, 0);
    lv_obj_set_flex_flow(main_cont, LV_FLEX_FLOW_COLUMN);

    lv_obj_t * cont = lv_obj_create(main_cont);
    lv_obj_remove_style_all(cont);
    lv_obj_set_flex_grow(cont, 1);
    lv_obj_set_style_pad_all(cont, 8, 0);
    lv_obj_set_width(cont, lv_pct(100));
    lv_subject_add_observer_obj(&current_tab_subject, cont_observer_cb, cont, NULL);
    lv_obj_set_scroll_dir(cont, LV_DIR_VER);

    lv_obj_t * footer = lv_obj_create(main_cont);
    lv_obj_remove_style_all(footer);
    lv_obj_set_style_pad_column(footer, 8, 0);
    lv_obj_set_style_pad_all(footer, 8, 0);
    lv_obj_set_flex_flow(footer, LV_FLEX_FLOW_ROW);
    lv_obj_set_size(footer, lv_pct(100), 60);
    lv_obj_align(footer, LV_ALIGN_BOTTOM_MID, 0, 0);

    btn_create("First");
    btn_create("Second");
    btn_create("Third");

    lv_obj_t * indicator = lv_obj_create(footer);
    lv_obj_remove_style(indicator, NULL, 0);
    lv_obj_set_style_bg_opa(indicator, LV_OPA_40, 0);
    lv_subject_add_observer_obj(&current_tab_subject, indicator_observer_cb, indicator, NULL);
    lv_obj_set_height(indicator, 10);
    lv_obj_align(indicator, LV_ALIGN_BOTTOM_LEFT, 0, 0);
    lv_obj_add_flag(indicator, LV_OBJ_FLAG_IGNORE_LAYOUT);

    /*Be sure the indicator has the correct size*/
    lv_obj_update_layout(indicator);
    lv_subject_notify(&current_tab_subject);
}

static int32_t anim_get_x_cb(lv_anim_t * a)
{
    return lv_obj_get_x_aligned((lv_obj_t *) a->var);
}

static void anim_set_x_cb(void * obj, int32_t v)
{
    lv_obj_set_x((lv_obj_t *) obj, v);
}

static void cont_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    int32_t prev_v = lv_subject_get_previous_int(subject);
    int32_t cur_v = lv_subject_get_int(subject);
    lv_obj_t * cont = (lv_obj_t *) lv_observer_get_target(observer);

    /*Animate out the previous content*/
    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_duration(&a, 300);
    lv_anim_set_path_cb(&a, lv_anim_path_ease_in_out);
    lv_anim_set_exec_cb(&a, anim_set_x_cb);
    lv_anim_set_get_value_cb(&a, anim_get_x_cb);
    lv_anim_set_completed_cb(&a, lv_obj_delete_anim_completed_cb);

    uint32_t i;
    uint32_t delay = 0;
    uint32_t child_cnt_prev = lv_obj_get_child_count(cont);
    for(i = 0; i < child_cnt_prev; i++) {
        lv_obj_t * child = lv_obj_get_child(cont, i);
        lv_anim_set_var(&a, child);
        if(prev_v < cur_v) {
            lv_anim_set_values(&a, 0, -20);
        }
        else {
            lv_anim_set_values(&a, 0, 20);
        }
        lv_anim_set_delay(&a, delay);
        lv_anim_start(&a);
        lv_obj_fade_out(child, 200, delay);
        delay += 50;
    }

    /*Create the widgets according to the current value*/
    if(cur_v == 0) {
        for(i = 0; i < 4; i++) {
            lv_obj_t * slider = lv_slider_create(cont);
            lv_slider_bind_value(slider, &slider_subject[i]);
            lv_obj_align(slider, LV_ALIGN_TOP_MID, 0, 10 + i * 30);
        }
    }
    if(cur_v == 1) {
        for(i = 0; i < 3; i++) {
            lv_obj_t * dropdown = lv_dropdown_create(cont);
            lv_dropdown_bind_value(dropdown, &dropdown_subject[i]);
            lv_obj_align(dropdown, LV_ALIGN_TOP_MID, 0, i * 50);
        }
    }
    if(cur_v == 2) {
        for(i = 0; i < 2; i++) {
            lv_obj_t * roller = lv_roller_create(cont);
        }
    }
}
```

(continues on next page)

(continua dalla pagina precedente)

```

        lv_roller_bind_value(roller, &roller_subject[i]);
        lv_obj_align(roller, LV_ALIGN_CENTER, - 80 + i * 160, 0);
    }

    /*Animate in the new widgets*/
    lv_anim_set_completed_cb(&a, NULL);
    for(i = child_cnt_prev; i < lv_obj_get_child_count(cont); i++) {
        lv_obj_t * child = lv_obj_get_child(cont, i);
        lv_anim_set_var(&a, child);
        if(prev_v < cur_v) {
            lv_anim_set_values(&a, 20, 0);
        }
        else {
            lv_anim_set_values(&a, -20, 0);
        }
        lv_anim_set_delay(&a, delay);
        lv_anim_start(&a);
        lv_obj_fade_in(child, 200, delay);
        delay += 50;
    }
}

static void btn_create(lv_obj_t * parent, const char * text)
{
    lv_obj_t * btn = lv_button_create(parent);
    lv_obj_set.flex_grow(btn, 1);
    lv_obj_set_height(btn, lv_pct(100));
    lv_obj_set_style_radius(btn, 0, 0);
    lv_subject_add_observer_obj(&current_tab_subject, btn_observer_cb, btn, NULL);
    lv_obj_add_event_cb(btn, btn_click_event_cb, LV_EVENT_CLICKED, NULL);

    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, text);
    lv_obj_center(label);
}

static void btn_click_event_cb(lv_event_t * e)
{
    lv_obj_t * btn = lv_event_get_target_obj(e);
    uint32_t idx = lv_obj_get_index(btn);
    lv_subject_set_int(&current_tab_subject, idx);
}

static void btn_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    int32_t prev_v = lv_subject_get_previous_int(subject);
    int32_t cur_v = lv_subject_get_int(subject);

    lv_obj_t * btn = (lv_obj_t *) lv_observer_get_target(observer);
    int32_t idx = (int32_t)lv_obj_get_index(btn);

    if(idx == prev_v) lv_obj_remove_state(btn, LV_STATE_CHECKED);
    if(idx == cur_v) lv_obj_add_state(btn, LV_STATE_CHECKED);
}

static void indicator_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    int32_t cur_v = lv_subject_get_int(subject);
    lv_obj_t * indicator = (lv_obj_t *) lv_observer_get_target(observer);

    lv_obj_t * footer = lv_obj_get_parent(indicator);
    lv_obj_t * btn_act = lv_obj_get_child(footer, cur_v);
    lv_obj_set_width(indicator, lv_obj_get_width(btn_act));

    lv_anim_t a;
    lv_anim_init(&a);
    lv_anim_set_exec_cb(&a, anim_set_x_cb);
    lv_anim_set_duration(&a, 300);
    lv_anim_set_path_cb(&a, lv_anim_path_ease_in_out);
    lv_anim_set_var(&a, indicator);
    lv_anim_set_values(&a, lv_obj_get_x(indicator), lv_obj_get_x(btn_act));
    lv_anim_start(&a);
}

#endif

```

## Firmware update process

```

#include "../../../lv_examples.h"
#ifndef LV_USE_OBSERVER || !LV_USE_ARC || !LV_USE_LABEL || !LV_USE_BUTTON || !LV_USE_SPINNER || !LV_BUILD_EXAMPLES
typedef enum {
    FW_UPDATE_STATE_IDLE,
    FW_UPDATE_STATE_CONNECTING,
    FW_UPDATE_STATE_CONNECTED,
    FW_UPDATE_STATE_DOWNLOADING,
    FW_UPDATE_STATE_CANCEL,
    FW_UPDATE_STATE_READY,
} lv_fw_update_state_t;

static void fw_upload_manager_observer_cb(lv_observer_t * observer, lv_subject_t * subject);
static void fw_update_btn_clicked_event_cb(lv_event_t * e);
static void fw_update_close_event_cb(lv_event_t * e);
static void fw_update_win_observer_cb(lv_observer_t * observer, lv_subject_t * subject);

static lv_subject_t fw_download_percent_subject;
static lv_subject_t fw_update_status_subject;

/**
 * Show how to handle a complete firmware update process with observers.
 * Normally it's hard to implement a firmware update process because in some cases
 * - the App needs to wait for the UI (wait for a button press)
 * - the UI needs to wait for the App (connecting or downloading)
 * With observers these complex mechanisms can be implemented a simple and clean way.
 */

```

(continues on next page)

(continua dalla pagina precedente)

```

void lv_example_observer_5(void)
{
    lv_subject_init_int(&fw_download_percent_subject, 0);
    lv_subject_init_int(&fw_update_status_subject, FW_UPDATE_STATE_IDLE);

    lv_subject_add_observer(&fw_update_status_subject, fw_upload_manager_observer_cb, NULL);

    /*Create start FW update button*/
    lv_obj_t * btn = lv_button_create(lv_screen_active());
    lv_obj_add_event_cb(btn, fw_update_btn_clicked_event_cb, LV_EVENT_CLICKED, NULL);
    lv_obj_center(btn);
    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, "Firmware update");
}

static void fw_update_btn_clicked_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);
    lv_obj_t * win = lv_win_create(lv_screen_active());
    lv_obj_set_size(win, lv_pct(90), lv_pct(90));
    lv_obj_set_height(lv_win_get_header(win), 40);
    lv_obj_set_style_radius(win, 8, 0);
    lv_obj_set_style_shadow_width(win, 24, 0);
    lv_obj_set_style_shadow_offset_x(win, 3, 0);
    lv_obj_set_style_shadow_offset_y(win, 3, 0);
    lv_obj_set_style_shadow_color(win, lv_color_hex3(0x888), 0);
    lv_win_add_title(win, "Firmware update");
    lv_obj_t * btn = lv_win_add_button(win, LV_SYMBOL_CLOSE, 40);
    lv_obj_add_event_cb(btn, fw_update_close_event_cb, LV_EVENT_CLICKED, NULL);
    lv_obj_center(win);

    lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_IDLE);
    lv_subject_add_observer_obj(&fw_update_status_subject, fw_update_win_observer_cb, win, NULL);
}

static void fw_update_close_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);
    lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_CANCEL);
}

static void restart_btn_click_event_cb(lv_event_t * e)
{
    lv_obj_t * win = (lv_obj_t *) lv_event_get_user_data(e);
    lv_obj_delete(win);
    lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_IDLE);
}

static void fw_update_win_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_obj_t * win = (lv_obj_t *) lv_observer_get_target(observer);
    lv_obj_t * cont = lv_win_get_content(win);
    lv_fw_update_state_t status = (lv_fw_update_state_t) lv_subject_get_int(&fw_update_status_subject);
    if(status == FW_UPDATE_STATE_IDLE) {
        lv_obj_clean(cont);
        lv_obj_t * spinner = lv_spinner_create(cont);
        lv_obj_center(spinner);
        lv_obj_set_size(spinner, 130, 130);

        lv_obj_t * label = lv_label_create(cont);
        lv_label_set_text(label, "Connecting");
        lv_obj_center(label);

        lv_subject_set_int(subject, FW_UPDATE_STATE_CONNECTING);
    }
    else if(status == FW_UPDATE_STATE_DOWNLOADING) {
        lv_obj_clean(cont);
        lv_obj_t * arc = lv_arc_create(cont);
        lv_arc_bind_value(arc, &fw_download_percent_subject);
        lv_obj_center(arc);
        lv_obj_set_size(arc, 130, 130);
        lv_obj_remove_flag(arc, LV_OBJ_FLAG_CLICKABLE);

        lv_obj_t * label = lv_label_create(cont);
        lv_label_bind_text(label, &fw_download_percent_subject, "%d %");
        lv_obj_center(label);
    }
    else if(status == FW_UPDATE_STATE_READY) {
        lv_obj_clean(cont);
        lv_obj_t * label = lv_label_create(cont);
        lv_label_set_text(label, "Firmware update is ready");
        lv_obj_align(label, LV_ALIGN_CENTER, 0, -20);

        lv_obj_t * btn = lv_button_create(cont);
        lv_obj_align(btn, LV_ALIGN_CENTER, 0, 20);
        lv_obj_add_event_cb(btn, restart_btn_click_event_cb, LV_EVENT_CLICKED, win);

        label = lv_label_create(btn);
        lv_label_set_text(label, "Restart");
    }
    else if(status == FW_UPDATE_STATE_CANCEL) {
        lv_obj_delete(win);
    }
}

static void connect_timer_cb(lv_timer_t * t)
{
    if(lv_subject_get_int(&fw_update_status_subject) != FW_UPDATE_STATE_CANCEL) {
        lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_CONNECTED);
    }
    lv_timer_delete(t);
}

static void download_timer_cb(lv_timer_t * t)
{
    if(lv_subject_get_int(&fw_update_status_subject) == FW_UPDATE_STATE_CANCEL) {
        lv_timer_delete(t);
        return;
    }

    int32_t v = lv_subject_get_int(&fw_download_percent_subject);
    if(v < 100) {

```

(continues on next page)

(continua dalla pagina precedente)

```

        lv_subject_set_int(&fw_download_percent_subject, v + 1);
    } else {
        lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_READY);
        lv_timer_delete(t);
    }
}

/**
 * Emulate connection and FW downloading by timers
 */
static void fw_upload_manager_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    LV_UNUSED(subject);
    LV_UNUSED(observer);

    lv_fw_update_state_t state = (lv_fw_update_state_t) lv_subject_get_int(&fw_update_status_subject);
    if(state == FW_UPDATE_STATE_CONNECTING) {
        lv_timer_create(connect_timer_cb, 2000, NULL);
    }
    else if(state == FW_UPDATE_STATE_CONNECTED) {
        lv_subject_set_int(&fw_download_percent_subject, 0);
        lv_subject_set_int(&fw_update_status_subject, FW_UPDATE_STATE_DOWNLOADING);
        lv_timer_create(download_timer_cb, 50, NULL);
    }
}
#endif

```

## Modular style update on theme change

```

#include "../../lv_examples.h"
#if LV_USE_OBSERVER && LV_USE_SLIDER && LV_USE_LABEL && LV_BUILD_EXAMPLES

typedef enum {
    THEME_MODE_LIGHT,
    THEME_MODE_DARK,
} lv_theme_mode_t;

static lv_obj_t * my_panel_create(lv_obj_t * parent);
static lv_obj_t * my_button_create(lv_obj_t * parent, const char * text, lv_event_cb_t event_cb);
static void switch_theme_event_cb(lv_event_t * e);

static lv_subject_t theme_subject;

/**
 * Change between light and dark mode
 */
void lv_example_observer_6(void)
{
    lv_subject_init_int(&theme_subject, THEME_MODE_DARK);

    lv_obj_t * panel1 = my_panel_create(lv_screen_active());
    lv_obj_set.flex_flow(panel1, LV_FLEX_FLOW_COLUMN);
    lv_obj.set.flex_align(panel1, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj.set.size(panel1, lv_pct(90), lv_pct(90));
    lv_obj_center(panel1);

    my_button_create(panel1, "Button 1", switch_theme_event_cb);
    my_button_create(panel1, "Button 2", switch_theme_event_cb);
    my_button_create(panel1, "Button 3", switch_theme_event_cb);
    my_button_create(panel1, "Button 4", switch_theme_event_cb);
    my_button_create(panel1, "Button 5", switch_theme_event_cb);
    my_button_create(panel1, "Button 6", switch_theme_event_cb);
    my_button_create(panel1, "Button 7", switch_theme_event_cb);
    my_button_create(panel1, "Button 8", switch_theme_event_cb);
    my_button_create(panel1, "Button 9", switch_theme_event_cb);
    my_button_create(panel1, "Button 10", switch_theme_event_cb);
}

static void switch_theme_event_cb(lv_event_t * e)
{
    LV_UNUSED(e);
    if(lv_subject_get_int(&theme_subject) == THEME_MODE_LIGHT) lv_subject_set_int(&theme_subject, THEME_MODE_DARK);
    else lv_subject_set_int(&theme_subject, THEME_MODE_LIGHT);
}

/*
 * my_panel.c
 * It would be a separate file with its own
 * local types, data, and functions
 */
typedef struct {
    lv_style_t style_main;
    lv_style_t style_scrollbar;
} lv_panel_styles_t;

static void my_panel_style_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    LV_UNUSED(subject);
    LV_UNUSED(observer);

    lv_theme_mode_t m = (lv_theme_mode_t) lv_subject_get_int(&theme_subject);
    lv_panel_styles_t * styles = (lv_panel_styles_t *) lv_observer_get_target(observer);
    if(m == THEME_MODE_LIGHT) {
        lv_style_set_bg_color(&styles->style_main, lv_color_hex3(0xffff));
        lv_style_set_shadow_color(&styles->style_main, lv_color_hex3(0x888));
        lv_style_set_text_color(&styles->style_main, lv_color_hex3(0x222));
        lv_style_set_bg_color(&styles->style_scrollbar, lv_color_hex3(0x888));
    }
    if(m == THEME_MODE_DARK) {
        lv_style_set_bg_color(&styles->style_main, lv_color_hex(0x040038));
        lv_style_set_shadow_color(&styles->style_main, lv_color_hex3(0xaa));
        lv_style_set_text_color(&styles->style_main, lv_color_hex3(0xee));
        lv_style_set_bg_color(&styles->style_scrollbar, lv_color_hex3(0xaa));
    }
}

```

(continues on next page)

(continua dalla pagina precedente)

```

lv_obj_report_style_change(&styles->style_main);
lv_obj_report_style_change(&styles->style_scrollbar);
}

static lv_obj_t * my_panel_create(lv_obj_t * parent)
{
    static bool init = false;
    static lv_panel_styles_t styles;
    if(init) {
        init = true;

        lv_style_init(&styles.style_main);
        lv_style_set_radius(&styles.style_main, 12);
        lv_style_set_bg_opa(&styles.style_main, LV_OPA_COVER);
        lv_style_set_shadow_width(&styles.style_main, 24);
        lv_style_set_shadow_offset_x(&styles.style_main, 4);
        lv_style_set_shadow_offset_y(&styles.style_main, 6);
        lv_style_set_pad_all(&styles.style_main, 12);
        lv_style_set_pad_gap(&styles.style_main, 16);

        lv_style_init(&styles.style_scrollbar);
        lv_style_set_width(&styles.style_scrollbar, 4);
        lv_style_set_radius(&styles.style_scrollbar, 2);
        lv_style_set_pad_right(&styles.style_scrollbar, 8);
        lv_style_set_pad_ver(&styles.style_scrollbar, 8);
        lv_style_set_bg_opa(&styles.style_scrollbar, LV_OPA_50);

        lv_subject_add_observer_with_target(&theme_subject, my_panel_style_observer_cb, &styles, NULL);
    }

    lv_obj_t * panel = lv_obj_create(parent);
    lv_obj_remove_style_all(panel);
    lv_obj_add_style(panel, &styles.style_main, 0);
    lv_obj_add_style(panel, &styles.style_scrollbar, LV_PART_SCROLLBAR);

    return panel;
}

/*-----
 * my_button.c
 *
 * It would be a separate file with its own
 * local types, data, and functions
 *-----*/
typedef struct {
    lv_style_t style_main;
    lv_style_t style_pressed;
} lv_button_styles_t;

static void my_button_style_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    LV_UNUSED(subject);
    LV_UNUSED(observer);

    lv_theme_mode_t m = (lv_theme_mode_t)lv_subject_get_int(&theme_subject);
    lv_button_styles_t * styles = (lv_button_styles_t *)lv_observer_get_target(observer);
    if(m == THEME_MODE_LIGHT) {
        lv_style_set_bg_color(&styles->style_main, lv_color_hex(0x3379de));
        lv_style_set_bg_grad_color(&styles->style_main, lv_color_hex(0xd249a5));
        lv_style_set_shadow_color(&styles->style_main, lv_color_hex(0x3379de));
        lv_style_set_text_color(&styles->style_main, lv_color_hex3(0xffff));

        lv_style_set_color_filter_opa(&styles->style_pressed, LV_OPA_70);
    }
    if(m == THEME_MODE_DARK) {
        lv_style_set_bg_color(&styles->style_main, lv_color_hex(0xde1382));
        lv_style_set_bg_grad_color(&styles->style_main, lv_color_hex(0xb0c72));
        lv_style_set_shadow_color(&styles->style_main, lv_color_hex(0x4b0c72));
        lv_style_set_text_color(&styles->style_main, lv_color_hex3(0xffff));

        lv_style_set_color_filter_opa(&styles->style_pressed, LV_OPA_30);
    }

    lv_obj_report_style_change(&styles->style_main);
    lv_obj_report_style_change(&styles->style_pressed);
}

static lv_obj_t * my_button_create(lv_obj_t * parent, const char * text, lv_event_cb_t event_cb)
{
    static bool init = false;
    static lv_button_styles_t styles;
    if(init) {
        init = true;

        lv_style_init(&styles.style_main);
        lv_style_set_radius(&styles.style_main, LV_RADIUS_CIRCLE);
        lv_style_set_bg_opa(&styles.style_main, LV_OPA_COVER);
        lv_style_set_bg_grad_dir(&styles.style_main, LV_GRAD_DIR_HOR);
        lv_style_set_shadow_width(&styles.style_main, 24);
        lv_style_set_shadow_offset_y(&styles.style_main, 6);
        lv_style_set_pad_hor(&styles.style_main, 32);
        lv_style_set_pad_ver(&styles.style_main, 12);

        lv_style_init(&styles.style_pressed);
        lv_style_set_color_filter_dsc(&styles.style_pressed, &lv_color_filter_shade);
        lv_subject_add_observer_with_target(&theme_subject, my_button_style_observer_cb, &styles, NULL);
    }

    lv_obj_t * btn = lv_button_create(parent);
    lv_obj_remove_style_all(btn);
    lv_obj_add_style(btn, &styles.style_main, 0);
    lv_obj_add_style(btn, &styles.style_pressed, LV_STATE_PRESSED);
    lv_obj_add_event_cb(btn, event_cb, LV_EVENT_CLICKED, NULL);

    lv_obj_t * label = lv_label_create(btn);
    lv_label_set_text(label, text);

    return btn;
}
#endif

```

## 9.11 Snapshot

Snapshot provides an API to take a snapshot image for an LVGL Widget together with its children. The image will look exactly like the Widget on the display.

### 9.11.1 Utilizzo

Simply call function `lv_snapshot_take(widget, color_format)` to generate the image descriptor which can be used as an Image Widget's image source using `lv_image_set_src()`.

Note, only following color formats are supported at this time:

- `LV_COLOR_FORMAT_RGB565`
- `LV_COLOR_FORMAT_RGB888`
- `LV_COLOR_FORMAT_XRGB8888`
- `LV_COLOR_FORMAT_ARGB8888`

### Freeing the Image

The memory `lv_snapshot_take()` uses is dynamically allocated using `lv_draw_buf_create()`. Use `lv_draw_buf_destroy()` to free the memory it allocated.

The snapshot image which is the draw buffer returned by `lv_snapshot_take()` normally won't be added to the cache because it can be drawn directly. So you don't need to invalidate the cache by calling `lv_image_cache_drop()` before destroying the draw buffer.

The below code snippet demonstrates correct use of `lv_snapshot_take()`:

```
void update_snapshot(lv_obj_t * widget, lv_obj_t * img_snapshot)
{
    lv_draw_buf_t* snapshot = (void*)lv_image_get_src(img_snapshot);
    if(snapshot) {
        lv_draw_buf_destroy(snapshot);
    }
    snapshot = lv_snapshot_take(widget, LV_COLOR_FORMAT_ARGB8888);
    lv_image_set_src(img_snapshot, snapshot);
}
```

### Using an Existing Buffer

If the snapshot needs to be updated repeatedly, or if the caller provides the draw buffer, use `lv_snapshot_take_to_draw_buf(widget, color_format, draw_buf)`. In this case, the caller is responsible for creating and destroying the draw buffer.

If snapshot is generated successfully, the image descriptor is updated, the image data will be stored to the provided `draw_buf`, and the function will return `LV_RESULT_OK`.

Note that snapshot may fail if the provided buffer is not large enough, which can happen if the Widget's size changes. It's recommended to use `lv_snapshot_reshape_draw_buf(widget, draw_buf)` to first ensure the buffer is large enough, and if it fails, destroy the existing draw buffer and call `lv_snapshot_take` directly.

### 9.11.2 Esempio

#### Simple snapshot example

```
#include "../../lv_examples.h"
#if LV_USE_SNAPSHOT && LV_BUILD_EXAMPLES

static void event_cb(lv_event_t * e)
{
    lv_obj_t * snapshot_obj = (lv_obj_t *) lv_event_get_user_data(e);
    lv_obj_t * img = lv_event_get_target_obj(e);

    if(snapshot_obj) {
        lv_draw_buf_t * snapshot = (lv_draw_buf_t *)lv_image_get_src(snapshot_obj);
        if(snapshot) {
            lv_draw_buf_destroy(snapshot);
        }
    }

    /*Update the snapshot, we know parent of object is the container.*/
    snapshot = lv_snapshot_take(lv_obj_get_parent(img), LV_COLOR_FORMAT_ARGB8888);
}
```

(continues on next page)

(continua dalla pagina precedente)

```

    if(snapshot == NULL)
        return;
    lv_image_set_src(snapshot_obj, snapshot);
}

void lv_example_snapshot_1(void)
{
    LV_IMAGE_DECLARE(img_star);
    lv_obj_t * root = lv_screen_active();
    lv_obj_set_style_bg_color(root, lv_palette_main(LV_PALETTE_LIGHT_BLUE), 0);

    /*Create an image object to show snapshot*/
    lv_obj_t * snapshot_obj = lv_image_create(root);
    lv_obj_set_style_bg_color(snapshot_obj, lv_palette_main(LV_PALETTE_PURPLE), 0);
    lv_obj_set_bg_opa(snapshot_obj, LV_OPA_100);
    lv_image_set_scale(snapshot_obj, 128);
    lv_image_set_rotation(snapshot_obj, 300);

    /*Create the container and its children*/
    lv_obj_t * container = lv_obj_create(root);

    lv_obj_center(container);
    lv_obj_set_size(container, 180, 180);
    lv_obj_set_flex_flow(container, LV_FLEX_FLOW_ROW_WRAP);
    lv_obj_set_flex_align(container, LV_FLEX_ALIGN_SPACE_EVENLY, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);
    lv_obj_set_style_radius(container, 50, 0);
    lv_obj_t * img;
    int i;
    for(i = 0; i < 4; i++) {
        img = lv_image_create(container);
        lv_image_set_src(img, &img_star);
        lv_obj_set_style_bg_color(img, lv_color_black(), 0);
        lv_obj_set_style_bg_opa(img, LV_OPA_COVER, 0);
        // lv_obj_set_style_transform_scale(img, 400, LV_STATE_PRESSED);
        lv_obj_add_flag(img, LV_OBJ_FLAG_CLICKABLE);
        lv_obj_add_event_cb(img, event_cb, LV_EVENT_PRESSED, snapshot_obj);
        lv_obj_add_event_cb(img, event_cb, LV_EVENT_RELEASED, snapshot_obj);
    }
}
#endif

```

### 9.11.3 API

*lv\_snapshot.h*

## 9.12 System Monitor (sysmon)

The System Monitor module provides real-time monitoring of system performance metrics directly on your display. It supports both performance monitoring (CPU usage and FPS) and memory monitoring (used memory and fragmentation).

### 9.12.1 Dipendenze

- Requires `LV_USE_LABEL = 1` in `lv_conf.h`
- Requires `LV_USE_OBSERVER = 1` in `lv_conf.h`
- Requires `LV_USE_SYSMON = 1` in `lv_conf.h`

### 9.12.2 Utilizzo

#### Configurazione

Enable in `lv_conf.h`:

```

/* Main sysmon enable */
#define LV_USE_SYSMON 1

/* Performance monitor (CPU% and FPS) */
#define LV_USE_PERF_MONITOR 1

/* Memory monitor (used + fragmentation) */
#define LV_USE_MEM_MONITOR 1

/* Optional: refresh period in ms */
#define LV_SYSMON_REFR_PERIOD_DEF 300

/* Optional: log to console instead of screen */
#define LV_USE_PERF_MONITOR_LOG_MODE 0

```

## Creating Monitors

```
/* Create generic monitor */
lv_obj_t * sysmon = lv_sysmon_create(lv_display_get_default());

/* Create performance monitor */
lv_sysmon_show_performance(NULL); /* NULL = default display */

/* Create memory monitor */
lv_sysmon_show_memory(NULL);
```

## Performance Monitor

Tracks: - FPS (Frames Per Second) - CPU usage (%) - Render time (ms) - Flush time (ms) - Self CPU usage (%) if enabled

Display format: .. code-block:

```
32 FPS, 45% CPU
8 ms
```

Where: - Line 1: FPS, Total CPU% - Line 2: Total time (Render | Flush)

### 9.12.3 Pause and Resume

`lv_sysmon_performance_pause(displ)` pauses the perf monitor.

`lv_sysmon_performance_resume(displ)` resumes the perf monitor.

## Memory Monitor

Displays: - Current memory usage (kB and %) - Peak memory usage (kB) - Fragmentation (%)

Display format: .. code-block:

```
24.8 kB (76%)
32.4 kB max, 18% frag.
```

## Positioning

Configure positions in `lv_conf.h`:

```
/* Top-right corner */
#define LV_USE_PERF_MONITOR_POS LV_ALIGN_TOP_RIGHT

/* Bottom-right corner */
#define LV_USE_MEM_MONITOR_POS LV_ALIGN_BOTTOM_RIGHT
```

### 9.12.4 Implementation Details

#### Initialization

Maintains: - Global memory monitor (`sysmon_mem`) - Per-display performance structures

#### Performance Measurement

Event-based collection:

Event	Measurement
LV_EVENT_REFR_START	Refresh interval start
LV_EVENT_REFR_READY	Record refresh duration
LV_EVENT_RENDER_START	Render time start
LV_EVENT_RENDER_READY	Record render duration
LV_EVENT_FLUSH_*	Measure flush operations

## Timers

- Performance: `perf_update_timer_cb`
- Memory: `mem_update_timer_cb`
- Default period: 300ms (`LV_SYSMON_REFR_PERIOD_DEF`)

## 9.13 UI Testing

### 9.13.1 Panoramica

The Test module provides functions to emulate clicks, key presses, encoder turns, time passing, and compare the UI with reference images.

These functions can be easily used in any test framework (such as Unity, GoogleTest, etc.), and assertions can be performed to check if, for example:

- A widget's value is different from the expected value after emulating user inputs.
- The values are incorrect after some time has passed.
- The screen's content is different from the reference image.
- Some event functions are not triggered.
- Ecc.

Note that it is assumed the tests are performed on a desktop or server environment, where there are no memory constraints.

### 9.13.2 Utilizzo

The Test module can be enabled by configuring `LV_USE_TEST` to a non-zero value, and it consists of the following components:

- Helpers
- Display emulation
- Input device emulation
- Screenshot comparison

#### Helpers

##### Time

To emulate elapsed time, two functions can be used:

1. `lv_test_wait(ms)`: Emulates that `ms` milliseconds have elapsed, but it also calls `lv_timer_handler` after each millisecond. This is useful to check if events (e.g., long press, long press repeat) and timers were triggered correctly over time.
2. `lv_test_fast_forward(ms)`: Jumps `ms` milliseconds ahead and calls `lv_timer_handler` only once at the end.

`lv_refr_now(NULL)` is called at the end of both functions to ensure that animations and widget coordinates are recalculated.

`lv_refr_now(NULL)` can also be called manually to force LVGL to refresh the emulated display.

## Memory Usage

If `LV_USE_STDLIB_MALLOC` is set to `LV_STDLIB_BUILTIN`, memory usage and memory leaks can be monitored.

```
size_t mem1 = lv_test_get_free_mem();
<create and delete items>
size_t mem2 = lv_test_get_free_mem();
if(mem1 != mem2) fail();
```

It might make sense to create and delete items in a loop many times and add a small tolerance to the memory leakage test. This might be needed due to potential memory fragmentation. Empirically, a tolerance of 32 bytes is recommended.

```
if(LV_ABS((int64_t)mem2 - (int64_t)mem1) > 32) fail();
```

## Display Emulation

By calling `lv_test_display_create(hor_res, ver_res)`, a dummy display can be created.

It functions like any other normal display, but its content exists only in memory.

When creating this display, the horizontal and vertical resolutions must be passed. Internally, a framebuffer will be allocated for this size, and `XRGB8888` color format will be set.

The resolution and color format can be changed at any time by calling `lv_display_set_resolution()` and `lv_display_set_color_format()`.

## Input Device Emulation

By calling `lv_test_indev_create_all()`, three test input devices will be created:

1. A pointer (for touch or mouse)
2. A keypad
3. An encoder

For example, this is how a scroll gesture can be emulated:

```
lv_test_mouse_move_to(20, 30);
lv_test_mouse_press();
lv_test_wait(20);
lv_test_mouse_move_by(0, 100);
lv_test_wait(20);
lv_test_mouse_release();
lv_test_wait(20);
```

It is recommended to add `lv_test_wait()` after user actions to ensure that the new state and coordinates are read and applied from the input device.

After that, the user can check if the given widget was really scrolled by getting the Y coordinate of a child.

```
int32_t y_start = lv_obj_get_y(child);
<scroll emulation>
int32_t y_end = lv_obj_get_y(child);
if(y_start + 100 != y_end) fail();
```

Please refer to `lv_test_indev.h` for the list of supported input device emulation functions.

## Screenshot Comparison

`bool lv_test_screenshot_compare(const char * fn_ref)` is a useful function to compare the content of the emulated display with reference PNG images.

This function works in a practical way:

- If the folder(s) referenced in `fn_ref` do not exist, they will be created automatically.
- If the reference image is not found, it will be created automatically from the rendered screen.
- If the comparison fails, an `<image_name>_err.png` file will be created with the rendered content next to the reference image.

- If the comparison fails, the X and Y coordinates of the first divergent pixel, along with the actual and expected colors, will also be printed.

The reference PNG images should have a **32-bit color format** and match the display size.

The test display's content will be converted to **XRGB8888** to simplify comparison with the reference images. The conversion is supported from the following formats (i.e., the test display should have a color format in this list):

- *LV\_COLOR\_FORMAT\_XRGB8888*
- *LV\_COLOR\_FORMAT\_ARGB8888*
- *LV\_COLOR\_FORMAT\_ARGB8888\_PREMULTIPLIED*
- *LV\_COLOR\_FORMAT\_RGB888*
- *LV\_COLOR\_FORMAT\_RGB565*
- *LV\_COLOR\_FORMAT\_RGB565\_SWAPPED*
- *LV\_COLOR\_FORMAT\_L8*
- *LV\_COLOR\_FORMAT\_AL88*
- *LV\_COLOR\_FORMAT\_I1*

To read and decode PNG images and to store the converted rendered image, a few MBs of RAM are dynamically allocated using the standard `malloc` (not `lv_malloc`).

The screenshot comparison uses `lodepng` which is built-in to LVGL and just needs to be enabled with `LV_USE_LODEPNG`.

To avoid making the entire Test module dependent on `lodepng`, screenshot comparison can be individually enabled by `LV_USE_TEST_SCREENSHOT_COMPARE`.

### 9.13.3 API

*lv\_test\_display.h*  
*lv\_test\_helpers.h*  
*lv\_test\_indev.h*  
*lv\_test\_indev\_gesture.h*  
*lv\_test\_private.h*  
*lv\_test\_screenshot\_compare.h*

## 9.14 Traslazione

### 9.14.1 Panoramica

LVGL supports two ways of handling translations:

- **lv\_i18n**: A comprehensive tool that extracts translatable strings from C files into YAML files, and generates C translation files from them. It also supports plural forms. See its README for details.
- **lv\_translation**: A simpler yet more flexible solution that allows adding translations statically or dynamically. This is the method documented here.

### 9.14.2 Add Translations

#### Static Translations

If most translations are known at compile time, they can be defined using string arrays:

```

static const char * languages[] = {"en", "de", "es", NULL};
static const char * tags[] = {"tiger", "lion", "rabbit", "elephant", NULL};
static const char * translations[] = {
    "The Tiger", "Der Tiger", "El Tigre",
    "The Lion", "Der Löwe", "El León",
    "The Rabbit", "Das Kaninchen", "El Conejo",
    "The Elephant", "Der Elefant", "El Elefante",
};

lv_translation_add_static(languages, tags, translations);

```

This method uses only a little extra RAM, as only the pointers to the strings are stored.

### Dynamic Translations

If translations are only available at runtime (e.g., from files, serial ports, or online sources), they can be added dynamically.

This approach involves memory allocation. See the example at the bottom of this page for reference.

### 9.14.3 Select a Language

Once translations are registered, use:

`lv_translation_set_language("language")`

to set the current language. The parameter must match one of the language names provided during registration.

### 9.14.4 Translate Strings

To retrieve a translation for a given tag, use:

- `lv_translation_get("tag")`
- or the shorthand: `lv_tr("tag")`

These return a translated string which can be used with widgets:

```

lv_label_set_text(label, lv_tr("settings"));
lv_dropdown_set_options(dd, lv_tr("color_list"));

```

### Fallbacks

If a tag exists but the translation for the selected language is missing the tag itself will be returned.

If the tag is not found at all, the tag itself will be used as a fallback as well.

### 9.14.5 Esempio

#### Simple translation example

```

#include "../../lv_examples.h"
#if LV_USE_TRANSLATION && LV_BUILD_EXAMPLES

static void add_static(void)
{
    static const char * languages[] = {"en", "de", "es", NULL};
    static const char * tags[] = {"tiger", "lion", "rabbit", "elephant", NULL};
    static const char * translations[] = {
        "The Tiger", "Der Tiger", "El Tigre",
        "The Lion", "Der Löwe", "El León",
        "The Rabbit", "Das Kaninchen", "El Conejo",
        "The Elephant", "Der Elefant", "El Elefante",
    };
    lv_translation_add_static(languages, tags, translations);
}

static void add_dynamic(void)
{
    lv_translation_pack_t * pack = lv_translation_add_dynamic();
    lv_translation_add_language(pack, "en");
    lv_translation_add_language(pack, "de");

    lv_translation_tag_dsc_t * tag;
    tag = lv_translation_add_tag(pack, "table");
    lv_translation_set_tag_translation(pack, tag, 0, "It's a table");
    lv_translation_set_tag_translation(pack, tag, 1, "Das ist ein Tisch");

    tag = lv_translation_add_tag(pack, "chair");
    lv_translation_set_tag_translation(pack, tag, 0, "It's a chair");
    lv_translation_set_tag_translation(pack, tag, 1, "Das ist ein Stuhl");
}

```

(continues on next page)

(continua dalla pagina precedente)

```
}

/**
 * Create and use translations
 */
void lv_example_translation_1(void)
{
    add_static();
    add_dynamic();

    lv_translation_set_language("de");

    lv_obj_t * label;

    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, lv_tr("tiger"));

    label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, lv_tr("chair"));
    lv_obj_set_y(label, 50);
}

#endif /*LV_USE_TRANSLATION && LV_BUILD_EXAMPLES*/
```

## 9.14.6 API

*lv\_translation.h*

*lv\_translation\_private.h*

# CAPITOLO 10

---

## 3rd-Party Libraries

---

### 10.1 Arduino ESP littlefs

LittleFS is a little fail-safe filesystem designed for microcontrollers and integrated in the Arduino framework when used with ESP32 and ESP8266.

For a detailed introduction, see:

- <https://github.com/esp8266/Arduino>
- <https://github.com/espressif/arduino-esp32>

#### 10.1.1 Utilizzo

Enable `LV_USE_FS_ARDUINO_ESP_LITTLEFS` and define a `LV_FS_ARDUINO_ESP_LITTLEFS LETTER` in `lv_conf.h`.

#### 10.1.2 API

### 10.2 Arduino SD

Enables reading from and writing to SD cards. Once an SD memory card is connected to the SPI interface of the Arduino or Genuino board, you can create files and read from, and write to them. You can also move through directories on the SD card.

For a detailed introduction, see:

- <https://www.arduino.cc/reference/en/libraries/sd/>

#### 10.2.1 Utilizzo

Enable `LV_USE_FS_ARDUINO_SD` and define a `LV_FS_ARDUINO_SD LETTER` in `lv_conf.h`. You will need to initialize the SD card before LVGL can use it (i.e. `SD.begin(0, SPI, 4000000)`).

## 10.2.2 API

# 10.3 Barcode

The LVGL Barcode utility enables you to generate Code-128 bar codes. It uses the `code128` library by [fthunleth](#).

## 10.3.1 Utilizzo

Set `LV_USE_BARCODE` to 1 in `lv_conf.h`.

Use `lv_barcode_create()` to create a barcode object, and use `lv_barcode_update()` to generate a barcode.

Call `lv_barcode_set_scale()` to adjust scaling, call `lv_barcode_set_dark_color()` and `lv_barcode_set_light_color()` to adjust colors. Call `lv_barcode_set_direction()` to set the barcode's orientation.

By default, `LV_BARCODE_ENCODING_CODE128_GS1` encoding is used and strips [FCN1] and spaces. Optionally use `lv_barcode_set_encoding()` to set `LV_BARCODE_ENCODING_CODE128_RAW`.

After any of a bar code's settings have changed, call `lv_barcode_update()` again to regenerate it.

## 10.3.2 Notes

- It is best not to manually set the width of the barcode, because when the width of the Widget is lower than the width of the barcode, the display will be incomplete due to truncation.
- The scale adjustment can only be an integer multiple, for example, `lv_barcode_set_scale(barcode, 2)` means 2x scaling.
- The direction setting can be `LV_DIR_HOR` or `LV_DIR_VER`.

## 10.3.3 Esempio

### Create a Barcode

```
#include "../../../lv_examples.h"
#if LV_USE_BARCODE && LV_BUILD_EXAMPLES

/**
 * Create a Barcode
 */
void lv_example_barcode_1(void)
{
    lv_color_t bg_color = lv_palette_lighten(LV_PALETTE_LIGHT_BLUE, 5);
    lv_color_t fg_color = lv_palette_darken(LV_PALETTE_BLUE, 4);

    lv_obj_t * barcode = lv_barcode_create(lv_screen_active());
    lv_obj_set_height(barcode, 50);
    lv_obj_center(barcode);

    /*Set color*/
    lv_barcode_set_dark_color(barcode, fg_color);
    lv_barcode_set_light_color(barcode, bg_color);

    /*Add a border with bg_color*/
    lv_obj_set_style_border_color(barcode, bg_color, 0);

    /*Set data*/
    lv_barcode_update(barcode, "https://lvgl.io");
}

#endif
```

## 10.3.4 API

`lv_barcode.h`

`lv_barcode_private.h`

## 10.4 BMP Decoder

This BMP Decoder utility allows you to use images from .BMP files in LVGL.

Library source: <https://github.com/caj-johnson/bmp-decoder>

Instead of loading the whole image at once, BMP pixels are read on demand, so using BMP images requires very little RAM.

If enabled in `lv_conf.h` by setting `LV_USE_BMP` to 1, LVGL will register a new image decoder automatically so BMP files can be directly used as image sources. Esempio:

```
lv_image_set_src(my_img, "S:/path/to/picture.bmp");
```

Note that, a *File System* (`lv_fs_drv`) driver needs to registered to open images from files. Follow the instructions in *File System* (`lv_fs_drv`).

### 10.4.1 Limitazioni

- Only BMP files are supported. BMP images as C arrays (`lv_image_dsc_t`) are not. This is because there is no practical differences between how the BMP files and LVGL's image format stores the image data.
- BMP files can be loaded only from .BMP files. If you want to store them in flash it's better to convert them to a C array with LVGL's image converter.
- The BMP file's color format needs to match the configured `LV_COLOR_DEPTH` of the display on which it will be rendered. You can use GIMP to save the image in the required format. Both RGB888 and ARGB888 works with `LV_COLOR_DEPTH 32`
- Color palettes are not supported.
- Because the whole image is not loaded, it cannot be zoomed or rotated.

### 10.4.2 Esempio

#### Open a BMP image from file

```
#include "../../lv_examples.h"
#if LV_USE_BMP && LV_BUILD_EXAMPLES

/**
 * Open a BMP file from a file
 */
void lv_example_bmp_1(void)
{
    lv_obj_t * img = lv_image_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_image_set_src(img, "A:/lvgl/examples/libs/bmp/example_32bit.bmp");
    lv_obj_center(img);
}

#endif
```

### 10.4.3 API

`lv_bmp.h`

## 10.5 FFmpeg Support

**FFmpeg** is a complete, cross-platform solution to record, convert and stream audio and video.

The FFmpeg is an LVGL extension that interfaces with the official FFmpeg library to help you add platform-independent recording, converting and streaming audio and video into your LVGL UI.

The set-up steps below are for Linux, but they can be adapted for other platforms.

For a detailed introduction, see: <https://www.ffmpeg.org>

## 10.5.1 Installing FFmpeg

```
sudo apt install libavformat-dev libavcodec-dev libswscale-dev libavutil-dev
```

Or download the FFmpeg library from [its official download page](#), then install it:

```
./configure --disable-all --disable-autodetect --disable-podpages --disable-asm --enable-avcodec --enable-avformat --enable-decoders --enable-encoders --enable-demuxers --enable-parsers --enable-protocol='file' --enable-swscale --enable-zlib
make
sudo make install
```

## 10.5.2 Adding FFmpeg to Your Project

To use the FFmpeg library in your project, you will need to link against these libraries:

### **libavformat**

part of FFmpeg library

### **libavcodec**

part of FFmpeg library

### **libavutil**

part of FFmpeg library

### **libsSCALE**

part of FFmpeg library

### **libm**

### **libz**

### **libpthread**

If you are using GCC-based toolchain, this can be taken care of by adding the following command-line options:

```
-lavformat -lavcodec -lavutil -lswscale -lm -lz -lpthread
```

## 10.5.3 Utilizzo

Set the `LV_USE_FFMPEG` in `lv_conf.h` to 1.

Also set `LV_FFMPEG_PLAYER_USE_LV_FS` in `lv_conf.h` to 1 if you want to integrate the LVGL [File System \(lv\\_fs\\_drv\)](#) extension into FFmpeg. This library can load videos and images. The LVGL file system will always be used when an image is loaded with `lv_image_set_src()` regardless of the value of `LV_FFMPEG_PLAYER_USE_LV_FS`.

See the examples below for how to correctly use this library.

## 10.5.4 Eventi

- `LV_EVENT_READY` Sent when playback is complete and auto-restart is not enabled.

Scoprire di più sugli [Eventi](#).

## 10.5.5 Esempi

### Decode image

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FFMPEG && LV_FFMPEG_PLAYER_USE_LV_FS

/** 
 * Open an image from a file
 */
void lv_example_ffmpeg_1(void)
{
    /*It always uses the LVGL filesystem abstraction (not the OS filesystem)
     *to open the image, unlike `lv_ffmpeg_player_set_src` which depends on
     *the setting of `LV_FFMPEG_PLAYER_USE_LV_FS`.*/
    lv_obj_t * img = lv_image_create(lv_screen_active());
    lv_image_set_src(img, "A:lvgl/examples/libs/ffmpeg/ffmpeg.png");
    lv_obj_center(img);
}
```

(continues on next page)

(continua dalla pagina precedente)

```
#else
void lv_example_ffmpeg_1(void)
{
    /*TODO
     *fallback for online examples*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "FFmpeg is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

## Decode video

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FFMPEG

#if LV_FFMPEG_PLAYER_USE_LV_FS
#define PATH_PREFIX "A:"
#else
#define PATH_PREFIX "./"
#endif

/**
 * Open a video from a file
 */
void lv_example_ffmpeg_2(void)
{
    /*birds.mp4 is downloaded from http://www.videezy.com (Free Stock Footage by Videezy!)
     *https://www.videezy.com/abstract/44864-silhouettes-of-birds-over-the-sunset*/
    /*It will use the LVGL filesystem abstraction (not the OS filesystem)
     *if 'LV_FFMPEG_PLAYER_USE_LV_FS' is set.*/
    lv_obj_t * player = lv_ffmpeg_player_create(lv_screen_active());
    lv_ffmpeg_player_set_src(player, PATH_PREFIX "lvgl/examples/libs/ffmpeg/birds.mp4");
    lv_ffmpeg_player_set_auto_restart(player, true);
    lv_ffmpeg_player_set_cmd(player, LV_FFMPEG_PLAYER_CMD_START);
    lv_obj_center(player);
}

#else

void lv_example_ffmpeg_2(void)
{
    /*TODO
     *fallback for online examples*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "FFmpeg is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

## 10.5.6 API

*lv\_ffmpeg.h*

*lv\_ffmpeg\_private.h*

## 10.6 FreeType Support

**FreeType** is a freely available software library to render fonts.

The LVGL FreeType extension is an interface to the FreeType library, enabling you to generate font bitmaps at run time from most vector- and bitmap-font file formats.

For a detailed introduction, see: <https://freetype.org/>.

### 10.6.1 Adding FreeType to Your Project

First, Download FreeType from the `freetype2` folder (and optionally `freetype-docs` and `freetype-deemos`) from its [official repository](#). (The latest version is recommended.)

If you haven't already done so, now is a good time to get familiar with setting up and configuring this library. The above website is a good place to start, as is the `README` file in the top directory of the version you downloaded.

There are two ways to use FreeType:

## For UNIX

For UNIX-like systems, the following is recommended to compile and install FreeType libraries.

- Enter the FreeType source code directory
- `make`
- `sudo make install`
- Add include path: `/usr/include/freetype2` (for GCC: `-I/usr/include/freetype2 -L/usr/local/lib`)
- Link against library: `freetype` (for GCC: `-L/usr/local/lib -lfreetype`)

## For Embedded Devices

For embedded devices, it is recommended to use the FreeType configuration files provided by LVGL:

- `lvgl/src/libs/freetype/ftmodule.h`
- `lvgl/src/libs/freetype/ftoption.h`

which only include the most commonly used modules and options, which is important to save limited FLASH space.

- Copy the FreeType source code to your project directory.
- Refer to the following **Makefile** for configuration:

```
# FreeType custom configuration header file
CFLAGS += -DFT2_BUILD_LIBRARY
CFLAGS += -DFT_CONFIG_MODULES_H=<lvgl/src/libs/freetype/ftmodule.h>
CFLAGS += -DFT_CONFIG_OPTIONS_H=<lvgl/src/libs/freetype/ftoption.h>

# FreeType include path
CFLAGS += -I<freetype/include>

# FreeType C source file
FT_CSRCS += freetype/src/base/ftbase.c
FT_CSRCS += freetype/src/base/ftbitmap.c
FT_CSRCS += freetype/src/base/ftdebug.c
FT_CSRCS += freetype/src/base/ftglyph.c
FT_CSRCS += freetype/src/base/ftinit.c
FT_CSRCS += freetype/src/base/ftstroke.c
FT_CSRCS += freetype/src/cache/ftcache.c
FT_CSRCS += freetype/src/gzip/ftgzip.c
FT_CSRCS += freetype/src/sfnt/sfnt.c
FT_CSRCS += freetype/src/smooth/smooth.c
FT_CSRCS += freetype/src/truetype/truetype.c
CSRCS += $(FT_CSRCS)
```

### 10.6.2 Utilizzo

Set `LV_USE_FREETYPE` in `lv_conf.h` to 1.

Cache configuration:

- `LV_FREETYPE_CACHE_FT_GLYPH_CNT` Maximum number of cached glyphs., etc.

By default, the FreeType extension doesn't use LVGL's file system. You can simply pass the path to the font as usual on your operating system or platform.

If you want FreeType to use LVGL's memory allocation and file system interface, you can enable `LV_FREETYPE_USE_LVGL_PORT` in `lv_conf.h`, convenient for unified management. If you do this, you will need to exclude the configured FreeType library's `ftsystem.c` file from being compiled, since LVGL's `lv_ftsystem.c` has custom versions of the functions defined therein.

The font style supports ***Italic*** and ****Bold**** fonts processed by software, and can be set by using following values where style values are required:

- `LV_FREETYPE_FONT_STYLE_NORMAL`: Default style.
- `LV_FREETYPE_FONT_STYLE_ITALIC`: Italic style.
- `LV_FREETYPE_FONT_STYLE_BOLD`: Bold style.

These values can be combined, e.g. `LV_FREETYPE_FONT_STYLE_BOLD | LV_FREETYPE_FONT_STYLE_ITALIC`.

The FreeType extension also supports colored bitmap glyphs such as emojis. Note that only bitmaps are supported at this time. Colored vector graphics cannot be rendered. An example on how to draw a colored bitmap glyph is shown below.

Use the `lv_freetype_font_create()` function to create a font. To delete a font, use `lv_freetype_font_delete()`. For more detailed usage, please refer to the example code below.

### Ulteriori Informazioni

- FreeType tutorial
- LVGL's *Adding a New Font*

Rendering vector fonts is supported with VGLite or ThorVG, when using vector fonts with ThorVG, it is possible to set a letter outline of a different color.

This is achieved by setting the style attributes with the `lv_style_set_text_outline_width()` and `lv_style_set_text_outline_color()` functions

You will have to account for the increased width and height of letter due to the added letter outline, to avoid letters overlapping space them out using `lv_style_set_text_letter_space()`

To use vector fonts with ThorVG, you will have to enable `LV_USE_VECTOR_GRAPHICS` in `lv_conf.h`

### Nota

This feature is currently experimental, there are clipping issues especially when using large font sizes.

See the `lv_example_freetype_2_vector_font()` function for a usage example

## 10.6.3 Esempi

### Create a font with FreeType

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FREETYPE

#ifndef PATH_PREFIX
#define PATH_PREFIX "A:"
#endif

/* Load a font with FreeType */
void lv_example_freetype_1(void)
{
    /*Create a font*/
    lv_font_t * font = lv_freetype_font_create(PATH_PREFIX "lvgl/examples/libs/freetype/Lato-Regular.ttf",
                                                LV_FREETYPE_FONT_RENDER_MODE_BITMAP,
                                                24,
                                                LV_FREETYPE_FONT_STYLE_NORMAL);

    if(!font) {
        LV_LOG_ERROR("freetype font create failed.");
        return;
    }

    /*Create style with the new font*/
    static lv_style_t style;
    lv_style_init(&style);
    lv_style_set_text_font(&style, font);
    lv_style_set_text_align(&style, LV_TEXT_ALIGN_CENTER);

    /*Create a label with the new style*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_add_style(label, &style, 0);
    lv_label_set_text(label, "Hello world\nI'm a font created with FreeType");
    lv_obj_center(label);
}

#else

void lv_example_freetype_1(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "FreeType is not installed");
    lv_obj_center(label);
}
#endif
#endif
```

(continues on next page)

(continua dalla pagina precedente)

```
}
```

## Use a bitmap font to draw Emojis using FreeType

(continues on next page)

(continua dalla pagina precedente)

```

void lv_example_freetype_2_vector_font(uint32_t font_size, uint32_t border_width)
{
    /*TODO
     *fallback for online examples*/
    LV_UNUSED(font_size);
    LV_UNUSED(border_width);

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "FreeType is not installed");
    lv_obj_center(label);
}

void lv_example_freetype_2(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "FreeType is not installed");
    lv_obj_center(label);
}

#endif
#endif

```

## Freetype font kerning

```

#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_FREETYPE

#if LV_FREETYPE_USE_LVGL_PORT
#define PATH_PREFIX "A:"
#else
#define PATH_PREFIX "./"
#endif

#define KERNING_TEST_TEXT "AVAWAY,ToTaTe"

static void create_label(lv_font_kerning_t kerning, int32_t y_ofs, const char * text)
{
    lv_font_info_t font_info;
    lv_freetype_init_font_info(&font_info);
    font_info.name = PATH_PREFIX "lvgl/examples/libs/freetype/Lato-Regular.ttf";
    font_info.size = 32;
    font_info.kerning = kerning;

    lv_font_t * font = lv_freetype_font_create_with_info(&font_info);
    if(!font) {
        LV_LOG_ERROR("freetype font create failed.");
        return;
    }

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_set_style_text_font(label, font, 0);
    lv_label_set_text(label, text);
    lv_obj_align(label, LV_ALIGN_CENTER, 0, y_ofs);
}

/**
 * FreeType kerning example
 */
void lv_example_freetype_3(void)
{
    create_label(LV_FONT_KERNING_NONE, -32, "Kerning None: " KERNING_TEST_TEXT);
    create_label(LV_FONT_KERNING_NORMAL, 32, "Kerning Normal: " KERNING_TEST_TEXT);
}
#endif

void lv_example_freetype_3(void)
{
    /*TODO
     *fallback for online examples*/

    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "FreeType is not installed");
    lv_obj_center(label);
}

#endif
#endif

```

## 10.6.4 API

*lv\_freetype.h*

*lv\_freetype\_private.h*

## 10.7 File System Interfaces

LVGL's *File System (lv\_fs\_drv)* module provides an abstraction that enables you to attach any type of file system for LVGL's use. File systems already supported are:

- **FATFS**
- STDIO (Linux and Windows using C standard function .e.g **fopen**, **fread**)

- POSIX (Linux and Windows using POSIX function .e.g `open`, `read`)
- WIN32 (Windows using Win32 API function .e.g `CreateFileA`, `ReadFile`)
- MEMFS (read a file from a memory buffer)
- LITTLEFS (a little fail-safe filesystem designed for microcontrollers)
- Arduino ESP LITTLEFS (a little fail-safe filesystem designed for Arduino ESP)
- Arduino SD (allows for reading from and writing to SD cards)

You still need to provide the drivers and libraries, this extension provides only the bridge between LVGL and these file systems.

### 10.7.1 Utilizzo

In `lv_conf.h` enable `LV_USE_FS_...` (by setting its value to 1) and assign an upper cased letter to `LV_FS_..._DRIVER LETTER` (e.g. 'S'). If more than one file system is enabled, each driver will need to have a unique driver-identifier letter. After that you can access files using that driver letter. Example with driver identifier letter 'S':

**Linux-like relative path**  
`"S:/path/to/file.txt"`

**Linux-like absolute path**  
`"S:/path/to/file.txt"`

**Windows-like relative path**  
`"S:C:/path/to/file.txt"`

**Windows-like absolute path**  
`"S:C:/path/to/file.txt"`

Do not confuse the driver-identifier letter with the Windows/DOS/FAT "drive letter", which is part of the path passed to the OS-level functions. For more details, see [Identifier Letters](#).

[Cached reading](#) is also supported if `LV_FS_..._CACHE_SIZE` is set to a non-zero value.

For further details, including how to create support for your own file system, see [File System \(`lv\_fs\_drv`\)](#).

## 10.8 GIF Decoder

**GIF Decoder** is an LVGL extension that enables you to use GIF images in your LVGL UI.

For a detailed introduction, see: <https://github.com/lecrum/gifdec> .

### 10.8.1 Utilizzo

Once enabled in `lv_conf.h` by setting `LV_USE_GIF` to 1, `lv_gif_create(parent)` can be used to create a gif widget.

`lv_gif_set_src(widget, src)` works very similarly to `lv_image_set_src()`. As source, it also accepts images as variables (`lv_image_dsc_t`) or files.

#### Converting GIF Files to C Arrays

To convert a GIF file to an array of bytes, use LVGL's online converter. Select "Raw" color format and "C array" Output format.

#### Using GIF Images from Files

Esempio:

```
lv_gif_set_src(widget, "S:/path/to/example.gif");
```

Note that, a file system driver needs to be registered to open images from files. To do so, follow the instructions in *File System (lv\_fs\_drv)*.

## 10.8.2 Memory Requirements

To decode and display a GIF animation the following amount of RAM (in bytes) is required for each of the following color depths:

- *LV\_COLOR\_DEPTH* 8:  $3 \times \text{image width} \times \text{image height}$
- *LV\_COLOR\_DEPTH* 16:  $4 \times \text{image width} \times \text{image height}$
- *LV\_COLOR\_DEPTH* 32:  $5 \times \text{image width} \times \text{image height}$

## 10.8.3 Esempio

### Open a GIF image from file and variable

```
#include "../../lv_examples.h"
#if LV_USE_GIF && LV_BUILD_EXAMPLES

/**
 * Open a GIF image from a file and a variable
 */
void lv_example_gif_1(void)
{
    LV_IMAGE_DECLARE(img_bulb_gif);
    lv_obj_t * img;

    img = lv_gif_create(lv_screen_active());
    lv_gif_set_src(img, &img_bulb_gif);
    lv_obj_align(img, LV_ALIGN_LEFT_MID, 20, 0);

    img = lv_gif_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_gif_set_src(img, "A:/lvgl/examples/libs/gif/bulb.gif");
    lv_obj_align(img, LV_ALIGN_RIGHT_MID, -20, 0);
}

#endif
```

## 10.8.4 API

*lv\_gif.h*

*lv\_gif\_private.h*

## 10.9 littlefs

**littlefs** is a little fail-safe filesystem library designed for microcontrollers.

The *lv\_fs\_littlefs* extension is an interface to the littlefs library.

For a detailed introduction, see: <https://github.com/littlefs-project/littlefs> .

## 10.9.1 Utilizzo

Set *LV\_USE\_FS\_LITTLEFS* in *lv\_conf.h* to 1 and define an upper-case letter (as a C character type) for *LV\_FS\_LITTLEFS LETTER* in the range ['A'..'Z'].

When enabled *lv\_littlefs\_set\_handler()* can be used to set up a mount point.

## 10.9.2 Esempio

```
#include "lfs.h"

// configuration of the filesystem is provided by this struct
const struct lfs_config cfg = {
    // block device operations
    .read = user_provided_block_device_read,
    .prog = user_provided_block_device_prog,
    .erase = user_provided_block_device_erase,
    .sync = user_provided_block_device_sync,
    // block device configuration
    .read_size = 16,
```

(continues on next page)

(continua dalla pagina precedente)

```
.prog_size = 16,
.block_size = 4096,
.block_count = 128,
.cache_size = 16,
.lookahead_size = 16,
.block_cycles = 500,
};

// mount the filesystem
int err = lfs_mount(&lfs, &cfg);

// reformat if we can't mount the filesystem
// this should only happen on the first boot
if (err) {
    lfs_format(&lfs, &cfg);
    lfs_mount(&lfs, &cfg);
}

lv_littlefs_set_handler(&lfs);
```

### 10.9.3 API

*lv\_fsdrv.h*

See also: [lvgl/src/libs/fsdrv/lv\\_fs\\_littlefs.c](#)

## 10.10 libjpeg-turbo Decoder

**libjpeg-turbo** is an LVGL interface to the libjpeg-turbo library --- a JPEG image codec that uses SIMD instructions to accelerate baseline JPEG compression and decompression on x86, x86-64, Arm, PowerPC, and MIPS systems, as well as progressive JPEG compression on x86, x86-64, and Arm systems.

On such systems, libjpeg-turbo is generally 2-6x as fast as libjpeg, all else being equal.

For a detailed introduction, see: <https://libjpeg-turbo.org>.

Library source: <https://github.com/libjpeg-turbo/libjpeg-turbo>

### 10.10.1 Install

```
sudo apt install libjpeg-turbo8-dev
```

### 10.10.2 Adding libjpeg-turbo to Your Project

Cmake:

```
find_package(JPEG REQUIRED)
include_directories(${JPEG_INCLUDE_DIR})
target_link_libraries(${PROJECT_NAME} PRIVATE ${JPEG_LIBRARIES})
```

### 10.10.3 Utilizzo

Set *LV\_USE\_LIBJPEG\_TURBO* in *lv\_conf.h* to 1.

See the examples below.

It should be noted that each image decoded needs to consume:

image width × image height × 3

bytes of RAM, and it needs to be combined with the *Image caching* feature to ensure that the memory usage is within a reasonable range.

### 10.10.4 Esempio

#### Load a JPG image

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_LIBJPEG_TURBO
/** Load a JPG image */
void lv_example_libjpeg_turbo_1(void)
{
    lv_obj_t * wp;

    wp = lv_image_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_image_set_src(wp, "A:lvgl/examples/libs/libjpeg_turbo/flower.jpg");
    lv_obj_center(wp);
}

#else
void lv_example_libjpeg_turbo_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "LibJPEG-Turbo is not installed");
    lv_obj_center(label);
}
#endif
#endif
```

## 10.10.5 API

*lv\_libjpeg\_turbo.h*

## 10.11 libpng Decoder

**libpng** is an LVGL interface to the official PNG reference library, which supports almost all PNG features, is extensible, and has been extensively tested for over 28 years.

For a detailed introduction, see: <http://www.libpng.org/pub/png/libpng.html>.

### 10.11.1 Install

```
sudo apt install libpng-dev
```

### 10.11.2 Adding libpng to Your Project

Cmake:

```
find_package(PNG REQUIRED)
include_directories(${PNG_INCLUDE_DIR})
target_link_libraries(${PROJECT_NAME} PRIVATE ${PNG_LIBRARIES})
```

### 10.11.3 Utilizzo

Set *LV\_USE\_LIBPNG* in *lv\_conf.h* to 1.

See the examples below.

It should be noted that each image of this decoder needs to consume

width × height × 4

bytes of RAM, and it needs to be combined with the *Image caching* feature to ensure that the memory usage is within a reasonable range. The decoded image is stored in RGBA pixel format.

### 10.11.4 Esempio

#### Open a PNG image from file and variable

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES

#if LV_USE_LIBPNG
```

(continues on next page)

(continua dalla pagina precedente)

```
/** Open a PNG image from a file */
void lv_example_libpng_1(void)
{
    LV_IMAGE_DECLARE(img_png_demo);
    lv_obj_t * img;

    img = lv_image_create(lv_screen_active());
    lv_image_set_src(img, &img_png_demo);
    lv_obj_align(img, LV_ALIGN_LEFT_MID, 10, 0);

    img = lv_image_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_image_set_src(img, "A:lvgl/examples/libs/libpng/png_demo.png");
    lv_obj_align(img, LV_ALIGN_RIGHT_MID, -10, 0);
}

#else

void lv_example_libpng_1(void)
{
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "LibPNG is not installed");
    lv_obj_center(label);
}

#endif
#endif
```

## 10.11.5 API

*lv\_libpng.h*

## 10.12 LodePNG Decoder

**LodePNG** is an LVGL interface to the LodePNG library --- a PNG encoder and decoder in C and C++, without any dependencies, giving you an alternate way to use PNG images in your LVGL UI.

For a detailed introduction, see: <https://github.com/lvandeve/lodepng>.

If enabled in `lv_conf.h` by setting `LV_USE_LODEPNG` to 1, LVGL will register a new image decoder automatically so PNG files can be used directly as an image source.

### Nota

A file system driver needs to be registered to open images from files. To do so, follow the instructions in *File System (lv\_fs\_drv)*.

The whole PNG image is decoded, so

$\text{width} \times \text{height} \times 4$

bytes of RAM is required from the LVGL heap. The decoded image is stored in RGBA pixel format.

Since it might take significant time to decode PNG images LVGL's *Image caching* feature can be useful.

### 10.12.1 Compressing PNG Files

PNG file format supports True color (24/32 bit), and 8-bit palette colors. Usually cliparts, drawings, icons and simple graphics are stored in PNG format, that do not use the whole color space, so it is possible to compress further the image by using 8-bit palette colors, instead of 24/32 bit True color format. Because embedded devices have limited (flash) storage, it is recommended to compress images.

One option is to use a free online PNG compressor site, for example Compress PNG: <https://compresspng.com/>

### 10.12.2 Esempio

## Open a PNG image from file and variable

```
#include "../../lv_examples.h"
#if LV_USE_LODEPNG && LV_USE_IMAGE && LV_BUILD_EXAMPLES

/**
 * Open a PNG image from a file and a variable
 */
void lv_example_lodepng_1(void)
{
    LV_IMAGE_DECLARE(img_wink_png);
    lv_obj_t * img;

    img = lv_image_create(lv_screen_active());
    lv_image_set_src(img, &img_wink_png);
    lv_obj_align(img, LV_ALIGN_LEFT_MID, 20, 0);

    img = lv_image_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_image_set_src(img, "A:lvgl/examples/libs/lodepng/wink.png");
    lv_obj_align(img, LV_ALIGN_RIGHT_MID, -20, 0);
}

#endif
```

### 10.12.3 API

*lv\_lodepng.h*

## 10.13 QR Code

The QR-Code-generator library by nayuki is a 3rd-party library that generates QR-Code bitmaps.

The lv\_qrcode LVGL extension is an interface to that library which implements a custom Widget that generates and displays QR Codes using the library.

### 10.13.1 Utilizzo

Enable `LV_USE_QRCODE` in `lv_conf.h` by setting its value to 1.

Use `lv_qrcode_create()` to create the QR-Code Widget. Then use `lv_qrcode_update()` to generate the QR Code on it.

If you need to re-modify the size and color, use `lv_qrcode_set_size()` and `lv_qrcode_set_dark_color()` or `lv_qrcode_set_light_color()` respectively, and then call `lv_qrcode_update()` again to update the QR Code.

### 10.13.2 Notes

- QR Codes with less data are smaller, but they are scaled by an integer value to best fit to the given size.

### 10.13.3 Esempio

#### Create a QR Code

```
#include "../../lv_examples.h"
#if LV_USE_QRCODE && LV_BUILD_EXAMPLES
#include <string.h>

/**
 * Create a QR Code
 */
void lv_example_qrcode_1(void)
{
    lv_color_t bg_color = lv_palette_lighten(LV_PALETTE_LIGHT_BLUE, 5);
    lv_color_t fg_color = lv_palette_darken(LV_PALETTE_BLUE, 4);

    lv_obj_t * qr = lv_qrcode_create(lv_screen_active());
    lv_qrcode_set_size(qr, 150);
    lv_qrcode_set_dark_color(qr, fg_color);
    lv_qrcode_set_light_color(qr, bg_color);

    /*Set data*/
    const char * data = "https://lvgl.io";
    lv_qrcode_update(qr, data, strlen(data));
    lv_obj_center(qr);

    /*Add a border with bg_color*/
    lv_obj_set_style_border_color(qr, bg_color, 0);
    lv_obj_set_style_border_width(qr, 5, 0);
}
```

## 10.13.4 API

*lv\_qrcode.h*

*lv\_qrcode\_private.h*

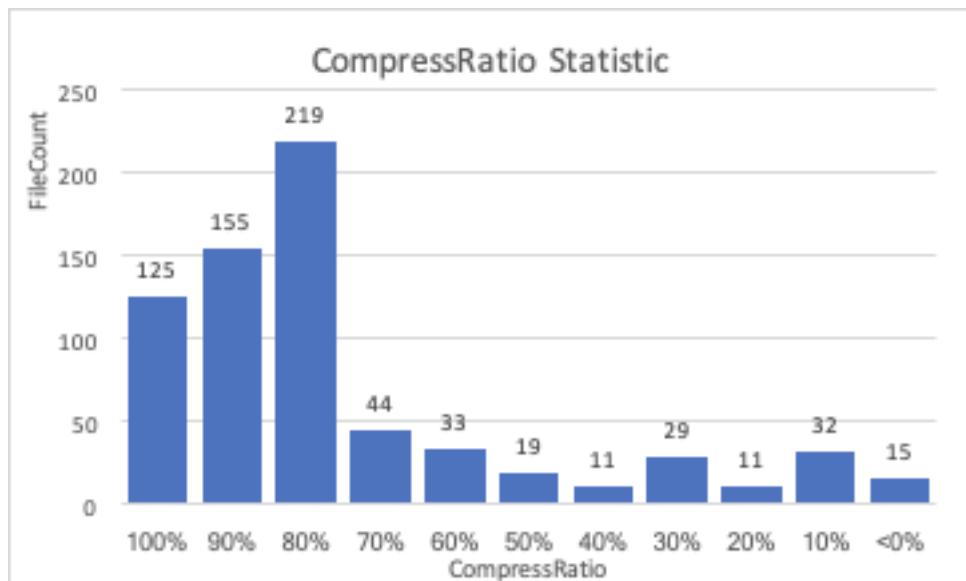
## 10.14 RLE Decompression

LVGL provides a custom RLE (Run-Length Encoding) decompression method. It can be used to reduce binary image size. The RLE compression is a lossless compression method.

The LVGL's built-in binary image decoder supports RLE-compressed images. The decoder supports both variable and file as image sources. The original binary data is directly decoded to RAM.

### 10.14.1 Benefits

Most screenshot and UI images (where there are a limited number of colors) can be compressed to save more than 70% space. The below statistics are from a watch project. It shows the file count of every compress level. For rare conditions, RLE compress may increase the file size if there's no large repetition in data.



### 10.14.2 Theory

The RLE algorithm is a simple compression algorithm that is based on the fact that for many adjacent pixels, the color is the same. The algorithm simply counts how many repeated pixels with the same color there are, and stores the count value and the color value. If the up-coming pixels are not repeated, it stores the non-repeat count value and original color values. For more details, the script used to compress the image can be found from `lvgl/script/LVGLImage.py`.

```
def rle_compress(self, data: bytearray, blksize: int, threshold=16):
    index = 0
    data_len = len(data)
    compressed_data = []
    while index < data_len:
        memview = memoryview(data)
        repeat_cnt = self.get_repeat_count(
            memview[index:], blksize)
        if repeat_cnt == 0:
            # done
            break
        elif repeat_cnt < threshold:
            nonrepeat_cnt = self.get_nonrepeat_count(
                memview[index:], blksize, threshold)
            ctrl_byte = uint8_t(nonrepeat_cnt | 0x80)
            compressed_data.append(ctrl_byte)
            compressed_data.append(
                memview[index: index + nonrepeat_cnt * blksize])
            index += nonrepeat_cnt * blksize
        else:
            ctrl_byte = uint8_t(repeat_cnt)
            compressed_data.append(ctrl_byte)
            compressed_data.append(memview[index: index + blksize])
```

(continues on next page)

(continua dalla pagina precedente)

```
index += repeat_cnt * blksize
return b"".join(compressed_data)
```

### 10.14.3 Utilizzo

To use the RLE Decoder, enable it in `lv_conf.h` configuration file by setting `LV_USE_RLE` to 1. The RLE image can then be used in the same way as other images.

```
lv_image_set_src(img, "path/to/image.rle");
```

### 10.14.4 Generating RLE Compressed Binary Images

An RLE image binary can be directly generated from another image using script `lvgl/script/LVGLImage.py`.

```
./scripts/LVGLImage.py --ofmt BIN --cf I8 --compress RLE cogwheel.png
```

This will decompress `cogwheel.png`, and then re-compress it using RLE and write the output to `./output/cogwheel.bin`.

### 10.14.5 API

`lv_rle.h`

## 10.15 Rlottie Player

### ⚠️ Avvertimento

Rlottie is deprecated. Consider using the [Lottie \(`lv\_lottie`\)](#) Widget instead.

The [Rlottie](#) animation player for LVGL is a 3rd-party extension for LVGL that allows playing Lottie animations in LVGL. It provides an interface to [Samsung/rlottie](#) library's C API. This Lottie player is not part of LVGL; it needs to be built separately.

### 10.15.1 Building Rlottie

To build Samsung's Rlottie, you will need a C++14-compatible compiler and optionally CMake 3.14 or higher.

To build on a desktop you can follow the instructions from Rlottie's README.

In the most basic case it looks like this:

```
mkdir rlottie_workdir
cd rlottie_workdir
git clone https://github.com/Samsung/rlottie.git
mkdir build
cd build
cmake ../rlottie
make -j
sudo make install
```

And finally add the `-lrlottie` flag to your linker.

On embedded systems you need to take care of integrating Rlottie to the given build system.

See the ESP-IDF example below.

### 10.15.2 Utilizzo

You can use animation from files or raw data (text). In either case first you need to enable `LV_USE_RLOTTIE` in `lv_conf.h` by setting its value to 1.

The `width` and `height` of the Widget is set in the `lv_rlottie_create_from_...()` function, and the animation will be scaled accordingly.

### Use Rlottie from File

To create a Lottie animation from a file, use:

```
lv_obj_t * lottie = lv_rlottie_create_from_file(parent, width, height, "path/to/lottie.json");
```

Note that, Rlottie uses the standard STUDIO C file API, so you can use the path "normally" and no LVGL specific driver letter is required.

### Use Rlottie from Raw String Data

`lv_example_rlottie_approve.c` contains an example animation in raw format. Invece di memorizzare la stringa JSON, viene memorizzato un array esadecimale per i seguenti motivi:

- avoid escaping " character in the JSON file
- some compilers don't support very long strings

`lvgl/scripts/filetohex.py` can be used to convert a Lottie file a hex array. Ad esempio:

```
./filetohex.py path/to/lottie.json --filter-character --null-terminate > out.txt
```

`--filter-character` filters out non-ASCII characters and `--null-terminate` makes sure that a trailing zero is appended to properly terminate the string.

To create an animation from raw data:

```
extern const uint8_t lottie_data[];
lv_obj_t* lottie = lv_rlottie_create_from_raw(parent, width, height, (const char *)lottie_data);
```

### 10.15.3 Getting Animations

Lottie is standard and popular format so you can find many animation files on the web. For example: <https://lottiefiles.com/>.

You can also create your own animations with Adobe After Effects or similar software.

### 10.15.4 Controlling Animations

LVGL provides two functions to control the animation mode: `lv_rlottie_set_play_mode()` and `lv_rlottie_set_current_frame()`. You'll combine your intentions when calling the first method, like in these examples:

```
lv_obj_t * lottie = lv_rlottie_create_from_file(scr, 128, 128, "test.json");
lv_obj_center(lottie);
// Pause to a specific frame
lv_rlottie_set_current_frame(lottie, 50);
lv_rlottie_set_play_mode(lottie, LV_RLOTTIE_CTRL_PAUSE); // The specified frame will be displayed and then the animation will pause
// Play backward and loop
lv_rlottie_set_play_mode(lottie, LV_RLOTTIE_CTRL_PLAY | LV_RLOTTIE_CTRL_BACKWARD | LV_RLOTTIE_CTRL_LOOP);
// Play forward once (no looping)
lv_rlottie_set_play_mode(lottie, LV_RLOTTIE_CTRL_PLAY | LV_RLOTTIE_CTRL_FORWARD);
```

The default animation mode is **play forward with loop**.

If you don't enable looping, a `LV_EVENT_READY` is sent when the animation cannot make more progress without looping.

To get the number of frames in an animation or the current frame index, you can cast the `lv_obj_t` instance to a `lv_rlottie_t` instance and inspect the `current_frame` and `total_frames` members.

## 10.15.5 ESP-IDF Example

### Background

Rlottie can be expensive to render on embedded hardware. Lottie animations tend to use a large amount of CPU time and can use large portions of RAM. This will vary from lottie to lottie but in general for best performance:

- Limit total # of frames in the animation
- Where possible, try to avoid bezier type animations
- Limit animation render size

If your ESP32 chip does not have SPIRAM you will face severe limitations in render size.

To give a better idea on this, lets assume you want to render a 240x320 lottie animation.

In order to pass initialization of the `lv_rlottie_t` object, you need 240x320x32/8 (307k) available memory. The latest ESP32-S3 has 256kb RAM available for this (before FreeRTOS and any other initialization starts taking chunks out). So while you can probably start to render a 50x50 animation without SPIRAM, PSRAM is highly recommended.

Additionally, while you might be able to pass initialization of the `lv_rlottie_t` object, as rlottie renders frame to frame, this consumes additional memory. A 30-frame animation that plays over 1 second probably has minimal issues, but a 300 frame animation playing over 10 seconds could very easily crash due to lack of memory as rlottie renders, depending on the complexity of the animation.

Rlottie will not compile for the IDF using the `-O2` compiler option at this time.

For stability in lottie animations, this author has found that they run best in the IDF when enabling `LV_MEM_CUSTOM` (using `stdlib.h`)

For all its faults, when running right-sized animations, they provide a wonderful utility to LVGL on embedded LCDs and can look really good when done properly.

When picking/designing a lottie animation consider the following limitations:

- Build the lottie animation to be sized for the intended size.
- It can scale/resize, but performance will be best when the base lottie size is as intended.
- Limit total number of frames, the longer the lottie animation is, the more memory it will consume for rendering (rlottie consumes IRAM for rendering).
- Build the lottie animation for the intended frame rate.
- Default lottie is 60fps, embedded LCDs likely won't go above 30fps.

### IDF Setup

Where the LVGL simulator uses the installed rlottie lib, the IDF works best when using rlottie as a submodule under the components directory.

```
cd 'your/project/directory'
git add submodule
git add submodule https://github.com/Samsung/rlottie.git ./components/rlottie/rlottie
git submodule update --init --recursive
```

Now, Rlottie is available as a component in the IDF, but it requires some additional changes and a CMakeLists file to tell the IDF how to compile.

### Rlottie Patch File

Rlottie relies on dynamic linking for an image loader lib. This needs to be disabled as the IDF doesn't play nice with dynamic linking.

A patch file is available in LVGL under: `/env_support/esp/rlottie/0001-changes-to-compile-with-esp-idf.patch`

Apply the patch file to your rlottie submodule.

## CMakeLists for IDF

An example CMakeLists file has been provided at `/env_support/esp/rottie/CMakeLists.txt`

Copy this CMakeLists file to '`your-project-directory'/components/rottie/`

In addition to the component CMakeLists file, you'll also need to tell your project level CMakeLists in your IDF project to require rottie:

```
REQUIRES "lvgl" "rottie"
```

From here, you should be able to use `lv_rottie` objects in your ESP-IDF project as any other widget in LVGL ESP examples. Please remember that these animations can be highly resource constrained and this does not guarantee that every animation will work.

## Additional Rottie Considerations in ESP-IDF

While unnecessary, removing the `rottie/rottie/example` folder can remove many un-needed files for this embedded LVGL application.

From here, you can use the relevant LVGL `lv_rottie...()` functions to create lottie animations in LVGL on embedded hardware!

Please note, that while lottie animations are capable of running on many ESP chips, below is recommended for best performance.

- ESP32-S3-WROOM-1-N16R8
  - 16mb quad spi flash
  - 8mb octal spi PSRAM
- IDF4.4 or higher

The Esp-box devkit meets this spec and <https://github.com/espressif/esp-box> is a great starting point to adding lottie animations.

You will need to enable `LV_USE_RLOTTIE` through `idf.py` menuconfig under LVGL component settings.

## Additional Changes to Make Use of SPIRAM

`lv_alloc / realloc` does not make use of SPIRAM. Given the high memory usage of lottie animations, it is recommended to shift as much out of internal DRAM into SPIRAM as possible. In order to do so, SPIRAM will need to be enabled in the menuconfig options for your given espressif chip.

There may be a better solution for this, but for the moment the recommendation is to make local modifications to the LVGL component in your espressif project. This is as simple as swapping `lv_alloc / lv_realloc` calls in `lv_rottie.c` with `heap_caps_malloc` (for IDF) with the appropriate `MALLOC_CAP` call --- for SPIRAM usage this is `MALLOC_CAP_SPIRAM`.

```
rottie->allocated_buf = heap_caps_malloc(allocated_buf_size+1, MALLOC_CAP_SPIRAM);
```

## 10.15.6 Esempio

### Load a Lottie animation from raw data

```
#include "../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_RLOTTIE

/** 
 * Load an lottie animation from flash
 */
void lv_example_rottie_1(void)
{
    extern const uint8_t lv_example_rottie_approve[];
    lv_obj_t * lottie = lv_rottie_create_from_raw(lv_screen_active(), 100, 100, (const char *)lv_example_rottie_approve);
    lv_obj_center(lottie);
}

#else
void lv_example_rottie_1(void)
{
```

(continues on next page)

(continua dalla pagina precedente)

```

/*TODO
 *fallback for online examples*/
lv_obj_t * label = lv_label_create(lv_screen_active());
lv_label_set_text(label, "Rlottie is not installed");
lv_obj_center(label);
}

#endif
#endif

```

### Load a Lottie animation from a file

```

#include "../../../lv_examples.h"
#if LV_BUILD_EXAMPLES
#if LV_USE_RLOTTIE

/**
 * Load an lottie animation from file
 */
void lv_example_rlottie_2(void)
{
    /*The rlottie library uses STUDIO file API, so there is no driver letter for LVGL*/
    lv_obj_t * lottie = lv_rlottie_create_from_file(lv_screen_active(), 100, 100,
                                                    "lvgl/examples/libs/rlottie/lv_example_rlottie_approve.json");
    lv_obj_center(lottie);
}

#else
void lv_example_rlottie_2(void)
{
    /*TODO
     *fallback for online examples*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_label_set_text(label, "Rlottie is not installed");
    lv_obj_center(label);
}

#endif
#endif

```

## 10.15.7 API

*lv\_rlottie.h*

*lv\_rlottie\_private.h*

## 10.16 SVG Support

The `lv_svg` extension provides makes it possible to use SVG images in your LVGL UI using the Scalable Vector Graphics (SVG) Tiny 1.2 Specification.

For a detailed introduction, see: <https://www.w3.org/TR/SVGTiny12/>

### 10.16.1 Utilizzo

Enable `LV_USE_SVG` in `lv_conf.h` by setting its value to 1.

See the examples below.

If you need support for SVG animation attribute parsing, you can set `LV_USE_SVG_ANIMATION` in `lv_conf.h` to 1.

### 10.16.2 Esempio

```

lv_svg_node_t * svg_doc;
const char* svg_data = "<svg><rect x=\"0\" y=\"0\" width=\"100\" height=\"100\"/></svg>";

/* Create an SVG DOM tree */
svg_doc = lv_svg_load_data(svg_data, svg_len);
...

/* Draw SVG image */
lv_draw_svg(layer, svg_doc);
...

/* Release the DOM tree */
lv_svg_node_delete(svg_doc);

```

`lv_image` also supports SVG images, For example:

```
lv_image_set_src(widget, "S:/path/to/example.svg");
```

### 10.16.3 API

*lv\_svg.h*  
*lv\_svg\_decoder.h*  
*lv\_svg\_parser.h*  
*lv\_svg\_render.h*  
*lv\_svg\_token.h*

## 10.17 Tiny TTF font engine

The `lv_tiny_ttf` extension allows using TrueType fonts in LVGL using the `stb_truetype` 3rd-Party Library.

For a detailed introduction, see: <https://github.com/nothings/stb>.

### 10.17.1 Utilizzo

When enabled in `lv_conf.h` by setting `LV_USE_TINY_TTF` to 1, `lv_tiny_ttf_create_data(data, data_size, font_size)` can be used to create a TTF font instance with the specified line height. You can then use that font anywhere `lv_font_t` is accepted.

By default, the TTF or OTF file must be embedded as an array, either in a header, or loaded into RAM in order to function.

However, if `LV_TINY_TTF_FILE_SUPPORT` is enabled (i.e. 1), `lv_tiny_ttf_create_file(path, font_size)` will also be available, allowing tiny\_ttf to stream from a file. The file must remain open the entire time the font is being used.

After a font is created, you can change the font size in pixels by using `lv_tiny_ttf_set_size(font, font_size)`.

By default, a font will cache data for up to 256 glyph elements to speed up rendering. This maximum can be changed by using `lv_tiny_ttf_create_data_ex(data, data_size, font_size, kerning, cache_size)` or `lv_tiny_ttf_create_file_ex(path, font_size, kerning, cache_size)` (when available). The cache size is indicated in number of entries. The `kerning` argument will be one of the `LV_FONT_KERNING_...` values, indicating whether to allow kerning, if supported, or disable.

### 10.17.2 Esempio

#### Open a font with Tiny TTF from data array

```
#include "../../lv_examples.h"
#if LV_USE_TINY_TTF && LV_BUILD_EXAMPLES

/***
 * Load a font with Tiny_TTF
 */
void lv_example_tiny_ttf_1(void)
{
    extern const uint8_t ubuntu_font[];
    extern const int ubuntu_font_size;

    /*Create style with the new font*/
    static lv_style_t style;
    lv_style_init(&style);
    lv_font_t * font = lv_tiny_ttf_create_data(ubuntu_font, ubuntu_font_size, 30);
    lv_style_set_text_font(&style, font);
    lv_style_set_text_align(&style, LV_TEXT_ALIGN_CENTER);

    /*Create a label with the new style*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_add_style(label, &style, 0);
    lv_label_set_text(label, "Hello world\nI'm a font\ncreated\nwith Tiny TTF");
    lv_obj_center(label);
}

#endif
```

## Load a font with Tiny\_TTF from file

```
#include "../../lv_examples.h"
#if LV_USE_TINY_TTF && LV_TINY_TTF_FILE_SUPPORT && LV_BUILD_EXAMPLES

/**
 * Load a font with Tiny_TTF from file
 */
void lv_example_tiny_ttf_2(void)
{
    /*Create style with the new font*/
    static lv_style_t style;
    lv_style_init(&style);
    lv_font_t * font = lv_tiny_ttf_create_file("A:lvgl/examples/libs/tiny_ttf/Ubuntu-Medium.ttf", 30);
    lv_style_set_text_font(&style, font);
    lv_style_set_text_align(&style, LV_TEXT_ALIGN_CENTER);

    /*Create a label with the new style*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_add_style(label, &style, 0);
    lv_label_set_text(label, "Hello world\nI'm a font\ncreated\nwith Tiny TTF");
    lv_obj_center(label);
}
#endif
```

## Change font size with Tiny\_TTF

```
#include "../../lv_examples.h"
#if LV_USE_TINY_TTF && LV_BUILD_EXAMPLES && LV_USE_OBSERVER

static void font_size_observer_cb(lv_observer_t * observer, lv_subject_t * subject);

static lv_subject_t subject_font;

/**
 * Change font size with Tiny_TTF
 */
void lv_example_tiny_ttf_3(void)
{
    extern const uint8_t ubuntu_font[];
    extern const int ubuntu_font_size;

    lv_subject_init_int(&subject_font, 25);

    /*Create style with the new font*/
    static lv_style_t style;
    lv_style_init(&style);
    lv_font_t * font = lv_tiny_ttf_create_data(ubuntu_font, ubuntu_font_size, 25);
    lv_style_set_text_font(&style, font);
    lv_style_set_text_align(&style, LV_TEXT_ALIGN_CENTER);

    lv_obj_t * slider = lv_slider_create(lv_screen_active());
    lv_obj_center(slider);
    lv_slider_set_range(slider, 5, 50);
    lv_obj_align(slider, LV_ALIGN_BOTTOM_MID, 0, -50);
    lv_slider_bind_value(slider, &subject_font);

    lv_obj_t * slider_label = lv_label_create(lv_screen_active());
    lv_obj_align_to(slider_label, slider, LV_ALIGN_OUT_BOTTOM_MID, 0, 10);
    lv_label_bind_text(slider_label, &subject_font, "%d");

    /*Create a label with the new style*/
    lv_obj_t * label = lv_label_create(lv_screen_active());
    lv_obj_add_style(label, &style, 0);
    lv_obj_set_size(label, LV_SIZE_CONTENT, LV_SIZE_CONTENT);
    lv_label_set_text(label, "Hello world!");
    lv_obj_center(label);

    lv_subject_add_observer(&subject_font, font_size_observer_cb, &style);
}

static void font_size_observer_cb(lv_observer_t * observer, lv_subject_t * subject)
{
    lv_style_t * style = (lv_style_t *) lv_observer_get_user_data(observer);
    lv_style_value_t v;
    lv_style_get_prop(style, LV_STYLE_TEXT_FONT, &v);
    lv_font_t * font = (lv_font_t *) v.ptr;
    int32_t size = lv_subject_get_int(subject);

    lv_tiny_ttf_set_size(font, size);

    lv_obj_report_style_change(style);
}
#endif
```

## 10.17.3 API

[lv\\_tiny\\_ttf.h](#)

## 10.18 Tiny JPEG Decompressor (TJpgDec)

**Tiny JPEG Decompressor** is an LVGL interface to the TJpgDec library --- a generic JPEG image decompressor module that highly optimized for small embedded systems. It works with very low memory consumption, so that it can be incorporated into tiny microcontrollers, such as AVR, 8051, PIC, Z80, Cortex-M0, etc..

For a detailed introduction, see: [TJpgDec](#).

## 10.18.1 Panoramica

Features and restrictions:

- JPEG is decoded in 8x8 tiles.
- Only baseline JPEG files are supported (no progressive JPEG support).
- Read from file and C array are implemented.
- Only the required portions of the JPEG images are decoded, therefore they cannot be zoomed or rotated.

## 10.18.2 Utilizzo

Set `LV_USE_TJPGD` to 1 in `lv_conf.h`. LVGL will register a new image decoder automatically so JPEG files can be used directly as image sources.

Ad esempio:

```
lv_image_set_src(my_img, "S:/path/to/picture.jpg");
```

### Nota

A file system driver needs to be registered to open images from files. To do so, follow the instructions in *File System (lv\_fs\_drv)*.

## 10.18.3 Converter

### Converting JPEG to C array

- Use lvgl online tool <https://lvgl.io/tools/imageconverter>
- Color format = RAW, output format = C Array

## 10.18.4 Esempio

### Load a JPG image

```
#include "../../lv_examples.h"
#if LV_USE_TJPGD && LV_BUILD_EXAMPLES

/** 
 * Load a JPG image
 */
void lv_example_tj jpgd_1(void)
{
    lv_obj_t * wp;

    wp = lv_image_create(lv_screen_active());
    /* Assuming a File system is attached to letter 'A'
     * E.g. set LV_USE_FS_STDIO 'A' in lv_conf.h */
    lv_image_set_src(wp, "A:/test_img_lvgl_logo.jpg");
    lv_obj_center(wp);
}

#endif
```

## 10.18.5 API

## Debugging

---

### 11.1 GDB Plug-In

#### 11.1.1 Debugging LVGL with GDB

To facilitate debugging LVGL with GDB, a GDB plugin is provided. This plugin can be found in the `lvgl/scripts/gdb` directory. The GDB plugin can be used with any target where GDB is available. For example, you can use it to debug a device connected to a PC via JLink, which provides a GDB server. Additionally, if your device crashes and you have a core dump, you can use GDB to analyze the core dump. To load the LVGL GDB plugin within GDB's command line, type the following command:

```
source lvgl/scripts/gdb/gdbinit.py
```

Example of usage:

```
(gdb) source lvgl/scripts/gdb/gdbinit.py
(gdb) dump obj -L 2
obj@0x60700000dd10 (0, 0, 799, 599)
tabview@0x608000204ca0 (0, 0, 799, 599)
    obj@0x607000025dad (0, 0, 799, 69)
    obj@0x607000025e80 (0, 70, 799, 599)
    obj@0x60700002bd70 (743, 543, 791, 591)
    btn@0x60700002c7f0 (747, 547, 787, 587)
keyboard@0x60d0000f7040 (0, 300, 799, 599)
dropdown-list@0x608000205420 (0, 0, 129, 129)
    label@0x60d0000f7ba0 (22, 22, 56, 39)
(gdb)
```

The plugin provides the following commands.

- `dump obj`: Dump the object tree.
- `info style`: Show the object's style.
- `info draw_unit`: Display all current drawing unit information.

#### 11.1.2 Dump obj tree

`dump obj`: Dump the object tree.

`dump obj -L 2`: Dump the object tree with a depth of 2.

`dump obj -a 0x60700000dd10`: Dump the object tree starting from the specified address.

### 11.1.3 Show obj's style

This command can dump the object's local style, since style value is a union, it's displayed in all possible formats.

`info style address_of_obj`: Show the object's style.

Esempio:

```
(gdb) info style 0x60700000dd10
32 = {num = 90, ptr = 0x5a, color = {blue = 90 '\Z', green = 0 '\000', red = 0 '\000'}}
158 = {num = 32767, ptr = 0x7fff, color = {blue = 255 '\377', green = 127 '\177', red = 0 '\000'}}
(gdb) p lv_global->disp_default->act_scr
$4 = (lv_obj_t *) 0x60700000dd10
(gdb) info style $4
32 = {num = 90, ptr = 0x5a, color = {blue = 90 '\Z', green = 0 '\000', red = 0 '\000'}}
158 = {num = 32767, ptr = 0x7fff, color = {blue = 255 '\377', green = 127 '\177', red = 0 '\000'}}
(gdb)
```

### 11.1.4 Connect to Debugger

This command provides the ability to connect and debug GDB Python Script using IDE.

Connect to PyCharm / VSCode / Eclipse (not supported yet)

`debugger -t pycharm`

`debugger -t vscode`

`debugger -t eclipse`

How to use it specifically, search `pydevd_pycharm / debugpy` for details.

## 11.2 Logging

LVGL has a built-in *Log* module to inform the user about what is happening in the library.

### 11.2.1 Log Level

To enable logging, set `LV_USE_LOG` in `lv_conf.h` and set `LV_LOG_LEVEL` to one of the following values:

- `LV_LOG_LEVEL_TRACE`: A lot of logs to give detailed information
- `LV_LOG_LEVEL_INFO`: Log important events
- `LV_LOG_LEVEL_WARN`: Log if something unwanted happened but didn't cause a problem
- `LV_LOG_LEVEL_ERROR`: Only critical issues, where the system may fail
- `LV_LOG_LEVEL_USER`: Only user messages
- `LV_LOG_LEVEL_NONE`: Do not log anything

The events which have a higher level than the set log level will be logged as well. E.g., if you set `LV_LOG_LEVEL_WARN`, errors will also be logged.

### 11.2.2 Printing Logs

#### Logging with printf

If your system supports `printf`, you just need to enable `LV_LOG_PRINTF` in `lv_conf.h` to send the logs with `printf`.

#### Custom log function

If you can't use `printf` or want to use a custom function to log, you can register a "logger" callback with `lv_log_register_print_cb()`.

Ad esempio:

```

void my_log_cb(lv_log_level_t level, const char * buf)
{
    serial_send(buf, strlen(buf));
}
...

lv_log_register_print_cb(my_log_cb);

```

### 11.2.3 Add Logs

You can also use the log module via the `LV_LOG_TRACE/INFO/WARN/ERROR/USER(text)` or `LV_LOG(text)` functions. Here:

- `LV_LOG_TRACE/INFO/WARN/ERROR/USER(text)` append the following information to your `text`
- Log Level
- `__FILE__`
- `__LINE__`
- `__func__`
- `LV_LOG(text)` is similar to `LV_LOG_USER` but has no extra information attached.

### 11.2.4 API

*lv\_log.h*

## 11.3 Profiler

As the complexity of the application increases, performance issues such as low FPS and frequent cache misses causing lag may arise. LVGL has internally set up some hooks for performance measurement to help developers analyze and locate performance issues.

### 11.3.1 Introduzione

LVGL has a built-in trace system to track and record the timestamps of important events that occur during runtime, such as rendering events and user input events. These event timestamps serve as important metrics for performance analysis.

The trace system has a configurable record buffer that stores the names of event functions and their timestamps. When the buffer is full, the trace system prints the log information through the provided user interface.

The output trace logs are formatted according to Android's `systrace` format and can be visualized using `Perfetto`.

### 11.3.2 Utilizzo

#### Configure profiler

To enable the profiler, set `LV_USE_PROFILER` in `lv_conf.h` and configure the following options:

1. Enable the built-in profiler functionality by setting `LV_USE_PROFILER_BUILTIN`.
2. Buffer configuration: Set the value of `LV_PROFILER_BUILTIN_BUF_SIZE` to configure the buffer size. A larger buffer can store more trace event information, reducing interference with rendering. However, it also results in higher memory consumption.
3. Timestamp configuration: LVGL uses the `lv_tick_get()` function with a precision of 1ms by default to obtain timestamps when events occur. Therefore, it cannot accurately measure intervals below 1ms. If your system environment can provide higher precision (e.g., 1us), you can configure the profiler as follows:
  - Recommended configuration in **UNIX** environments:

```
#include <sys/syscall.h>
#include <sys/types.h>
#include <time.h>
#include <unistd.h>

static uint64_t my_get_tick_cb(void)
{
    struct timespec ts;
    clock_gettime(CLOCK_MONOTONIC, &ts);
    return ts.tv_sec * 1000000000 + ts.tv_nsec;
}

static int my_get_tid_cb(void)
{
    return (int)syscall(SYS_gettid);
}

static int my_get_cpu_cb(void)
{
    int cpu_id = 0;
    syscall(SYS_getcpu, &cpu_id, NULL);
    return cpu_id;
}

void my_profiler_init(void)
{
    lv_profiler_builtin_config_t config;
    lv_profiler_builtin_config_init(&config);
    config.tick_per_sec = 1000000000; /* One second is equal to 1000000000 nanoseconds */
    config.tick_get_cb = my_get_tick_cb;
    config.tid_get_cb = my_get_tid_cb;
    config.cpu_get_cb = my_get_cpu_cb;
    lv_profiler_builtin_init(&config);
}
```

- Recommended configuration in Arduino environments:

```
static uint64_t my_get_tick_cb(void)
{
    /* Use the microsecond time stamp provided by Arduino */
    return micros();
}

void my_profiler_init(void)
{
    lv_profiler_builtin_config_t config;
    lv_profiler_builtin_config_init(&config);
    config.tick_per_sec = 1000000; /* One second is equal to 1000000 microseconds */
    config.tick_get_cb = my_get_tick_cb;
    lv_profiler_builtin_init(&config);
}
```

4. Log output configuration: LVGL uses the `LV_LOG()` interface by default to output trace information. If you want to use another interface to output log information (e.g., file stream), you can redirect the log output using the following code:

```
static void my_log_print_cb(const char * buf)
{
    printf("%s", buf);
}

void my_profiler_init(void)
{
    lv_profiler_builtin_config_t config;
    lv_profiler_builtin_config_init(&config);
    ... /* other configurations */
    config.flush_cb = my_log_print_cb;
    lv_profiler_builtin_init(&config);
}
```

## Run the test scenario

Run the UI scenario that you want to measure, such as scrolling a scrollable page up and down or entering/exiting an application.

## Process the logs

Save the output log as `my_trace.txt`, use `trace_filter.py` for filtering and preprocessing:

```
./lvgl/scripts/trace_filter.py my_trace.txt
```

O

```
python3 ./lvgl/scripts/trace_filter.py my_trace.txt
```

You will obtain a processed text file named `trace.systrace`, which roughly contains the following content:

```
# tracer: nop
#
LVGL-1 [0] 2892.002993: tracing_mark_write: B|1|lv_timer_handler
LVGL-1 [0] 2892.002993: tracing_mark_write: B|1|lv_display_refr_timer
LVGL-1 [0] 2892.003459: tracing_mark_write: B|1|refr_invalid_areas
```

(continues on next page)

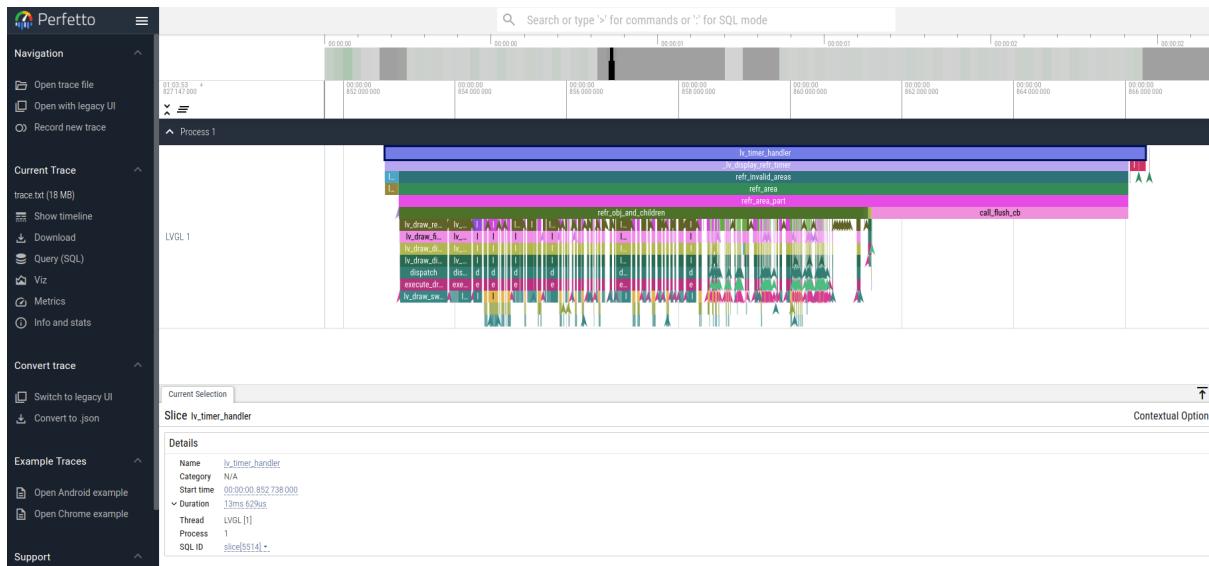
(continua dalla pagina precedente)

```
LVGL-1 [0] 2892.003461: tracing_mark_write: B|1|lv_draw_rect
LVGL-1 [0] 2892.00350: tracing_mark_write: E|1|lv_draw_rect
LVGL-1 [0] 2892.00352: tracing_mark_write: B|1|lv_draw_rect
LVGL-1 [0] 2892.00356: tracing_mark_write: E|1|lv_draw_rect
LVGL-1 [0] 2892.003560: tracing_mark_write: B|1|lv_draw_rect
LVGL-1 [0] 2892.003573: tracing_mark_write: E|1|lv_draw_rect
...
```

Import the processed `trace.systrace` file into **Perfetto** and wait for it to be parsed.

## Performance analysis

If the log parsing is successful, you will see the following screen:



In the Perfetto UI, use the **A** or **D** keys to pan the timeline horizontally and the **W** or **S** keys to zoom in or out on the timeline. Use the mouse to move the focus and click on functions on the timeline to observe their execution time.

### 11.3.3 Add Measurement Point

Users can add their own measured functions:

```
void my_function_1(void)
{
    LV_PROFILER_BEGIN;
    do_something();
    LV_PROFILER_END;
}

void my_function_2(void)
{
    LV_PROFILER_BEGIN_TAG("do_something_1");
    do_something_1();
    LV_PROFILER_END_TAG("do_something_1");

    LV_PROFILER_BEGIN_TAG("do_something_2");
    do_something_2();
    LV_PROFILER_END_TAG("do_something_2");
}
```

### 11.3.4 Custom profiler implementation

If you wish to use a profiler method provided by your operating system, you can modify the following configurations in `lv_conf.h`:

- `LV_PROFILER_INCLUDE`: Provides a header file for the profiler function.
- `LV_PROFILER_BEGIN`: Profiler start point function.
- `LV_PROFILER_END`: Profiler end point function.
- `LV_PROFILER_BEGIN_TAG`: Profiler start point function with custom tag.
- `LV_PROFILER_END_TAG`: Profiler end point function with custom tag.

Taking NuttX RTOS as an example:

```
#define LV_PROFILER_INCLUDE "nuttx/sched_note.h"
#define LV_PROFILER_BEGIN     sched_note_begin(NOTE_TAG_ALWAYS)
#define LV_PROFILER_END       sched_note_end(NOTE_TAG_ALWAYS)
#define LV_PROFILER_BEGIN_TAG(str) sched_note_beginex(NOTE_TAG_ALWAYS, str)
#define LV_PROFILER_END_TAG(str)  sched_note_endex(NOTE_TAG_ALWAYS, str)
```

## 11.3.5 FAQ

### Perfetto log parsing fails

Please check the completeness of the logs. If the logs are incomplete, it may be due to the following reasons:

1. Serial port reception errors caused by a high baud rate. You need to reduce the baud rate.
2. Data corruption caused by other thread logs inserted during the printing of trace logs. You need to disable the log output of other threads or refer to the configuration above to use a separate log output interface.
3. Make sure that the string passed in by `LV_PROFILER_BEGIN_TAG` or `LV_PROFILER_END_TAG` is not a local variable on the stack or a string in shared memory, because currently only the string address is recorded and the content is not copied.

### Function execution time displayed as 0s in Perfetto

If the function execution time is lower than the precision of the timestamps, this situation can occur. You can refer to the configuration instructions above to use a higher precision timestamp.

### Significant stuttering occurs during profiling

When the buffer used to store trace events becomes full, the profiler will output all the data in the buffer, which can cause UI blocking and stuttering during the output. You can optimize this by taking the following measures:

1. Increase the value of `LV_PROFILER_BUILTIN_BUF_SIZE`. A larger buffer can reduce the frequency of log printing, but it also consumes more memory.
2. Optimize the execution time of log printing functions, such as increasing the serial port baud rate or improving file writing speed.

### Trace logs are not being output

If the trace logs are not automatically printed when the buffer is not full, you can try the following methods to force the log output:

1. Reduce the value of `LV_PROFILER_BUILTIN_BUF_SIZE` to fill the buffer more quickly and trigger automatic printing.
2. Manually call or use a timer to call the `lv_profiler_builtin_flush()` function to force the log output.

## 11.4 VG-Lite Simulator

LVGL integra un simulatore VG-Lite basato su ThorVG. Its purpose is to simplify the debugging of VG-Lite adaptation and reduce the time spent debugging and locating problems on hardware devices.

It has been integrated into the CI automated compilation and testing process to ensure that the VG-Lite rendering backend can be fully tested after each PR modification.

### 11.4.1 How It Works

Sort out the APIs in the `vg_lite.h` header file provided by the vendor, re-implement the APIs using ThorVG, and simulate the same rendering images as the real hardware on the simulator.

### 11.4.2 Configurazione

1. Enable VG-Lite rendering backend, see [GPU Generica VG-Lite](#).
2. Enable ThorVG and turn on the configuration `LV_USE_THORVG_INTERNAL` or `LV_USE_THORVG_EXTERNAL`. It is recommended to use the internal ThorVG library to ensure uniform rendering results.
3. Enable `LV_USE_VG_LITE_THORVG` and set `LV_DRAW_BUF_ALIGN` to 64. The rest of the options can remain default. Make sure `LV_VG_LITE_USE_GPU_INIT` is enabled, because the thorvg drawing context needs to be initialized before it can be used.

# CAPITOLO 12

---

## Contribuire

---

### 12.1 Introduzione

Unirsi alla community di LVGL e lasciare il segno nella libreria!

Ci sono molti modi per contribuire a LVGL, anche se si è nuovi della libreria o anche solo un neofita della programmazione.

Potrebbe spaventare fare il primo passo, ma non c'è nulla di cui aver paura. Una community amichevole e disponibile ed in attesa. Si incontrano persone che la pensano in modo simile e si crea qualcosa di grande insieme.

Quindi, scopriamo quale opzione di contributo è più adatta ed è d'aiuto a partecipare allo sviluppo di LVGL!

### 12.2 Modi per Contribuire

- **Spargere la voce:** Condividere la propria esperienza con LVGL con gli amici o sui social media per aumentarne la visibilità.
- **Star LVGL** Dare una stella su [GitHub!](#) Aiuta molto a rendere LVGL più interessante per i nuovi arrivati.
- **Segnalare un Bug**\*: Aprire un [GitHub Issue](#) se qualcosa non funziona.
- **Unirsi al Nostro Forum** : Per incontrare altri sviluppatori e porre le proprie domande.
- **Raccontare le proprie idee:** Se si ritiene che manchi qualcosa in LVGL, saremo lieti di riceverlo tramite un [GitHub Issue](#)
- **Sviluppo di Funzionalità**\*: Aiuta a progettare o sviluppare una funzionalità. Vedere di seguito.

Le problematiche di media e larga scala vengono discusse nelle problematiche della [Feature Planning](#).

Una issue [problematica] può essere sviluppata quando tutte le domande nel "issue template" hanno ricevuto risposta e non ci sono obiezioni da parte di alcun membro principale.

Utilizziamo le etichette GitHub per mostrare lo stato e gli attributi delle problematiche e delle Pull Request. Per cercare opportunità di contributo, si può [Filtrare per queste etichette](#):

- **Simple:** Ottima scelta per iniziare con un contributo LVGL
- **PR needed:** Abbiamo esaminato il problema, ma deve ancora essere implementato
- **Review needed:** È stata aperta una Pull Request che necessita di revisione/test

## 12.3 Pull Request

Il merging di nuovo codice in `lvgl/lvgl` e in altri repository avviene tramite *Pull Request* (in breve PR). Una PR è una notifica del tipo "Ehi, ho apportato degli aggiornamenti al progetto. Ecco le modifiche, volendo, si possono aggiungere". Per fare ciò, c'è bisogno di una copia (chiamata fork) del progetto originale nel proprio account, apportare alcune modifiche e notificare al repository originale gli aggiornamenti. Si può vedere come appare su GitHub per LVGL qui: <https://github.com/lvgl/lvgl/pulls>.

Per aggiungere le modifiche, si possono modificare i file online su GitHub e inviare una nuova richiesta Pull da lì (consigliato per piccole modifiche) oppure aggiungere gli aggiornamenti nel proprio editor/IDE e usare `git` per pubblicare le modifiche (consigliato per aggiornamenti più complessi).

### 12.3.1 Da GitHub

1. Accedere al file da modificare.
2. Cliccare sul pulsante Edit nell'angolo in alto a destra.
3. Aggiungere le modifiche al file.
4. Aggiungere un messaggio di commit in fondo alla pagina.
5. Cliccare sul pulsante *Propose changes*.

### 12.3.2 Dalla Workstation Locale

Queste istruzioni descrivono il repository principale `lvgl`, ma funzionano allo stesso modo di qualsiasi repository Git remoto.

1. Eseguire il fork del repository `lvgl` repository. Per farlo, cliccare sul pulsante "Fork" nell'angolo in alto a destra. Il repository `lvgl` verrà "copiato" sul proprio account GitHub ([https://github.com/<YOUR\\_NAME>/lvgl](https://github.com/<YOUR_NAME>/lvgl))
2. Clonare il repository forkato.
3. Aggiungere le modifiche. Si può creare un *feature branch* dal branch `master` per gli aggiornamenti: `git checkout -b <new-feature-branch-name>`
4. Eseguire il commit e il push delle modifiche nel proprio repository `lvgl` forkato.
5. Creare una PR su GitHub dalla pagina del proprio repository `lvgl` forkato ([https://github.com/<YOUR\\_NAME>/lvgl/pulls](https://github.com/<YOUR_NAME>/lvgl/pulls)) cliccando sul pulsante "*New pull request*". Non dimenticare di selezionare il branch in cui sono state aggiunte le modifiche.
6. Impostare il branch di base in cui si desidera unire l'aggiornamento. Nel repository `lvgl`, sia le correzioni che le nuove funzionalità devono essere indirizzate al branch `master`.
7. Descrivere il contenuto dell'aggiornamento. Se applicabile, è benvenuto un codice di esempio.
8. Per apportare ulteriori modifiche, aggiornare il proprio repository `lvgl` forkato con nuovi commit. Appariranno automaticamente nella PR.

### 12.3.3 Formato dei Messaggi di Commit

Il formato dei messaggi di commit si ispira al Angular Commit Format.

È necessario utilizzare la seguente struttura:

```
<type>(<scope>): <subject>
  <--- blank line
<body>
  <--- blank line
<footer>
```

Possibili `<type>`:

- `fix` correzione di bug nel codice sorgente LVGL
- `feat` nuova funzionalità

- **arch** modifiche architetturali
- **perf** modifiche che influiscono sulle prestazioni
- **example** qualsiasi elemento relativo agli esempi (inclusi fix e nuovi esempi)
- **docs** qualsiasi elemento relativo alla documentazione (inclusi fix, formattazione e nuove pagine)
- **test** qualsiasi elemento relativo ai test (test nuovi e aggiornati o azioni di CI)
- **chore** qualsiasi modifica minore di formattazione o stile che renderebbe il changelog poco chiaro

**<scope>** è il nome del modulo, file o sottosistema interessato dal commit. Di solito è una sola parola e può essere scelta liberamente. Ad esempio `img`, `layout`, `txt`, `anim`. Lo scope può essere omesso.

**<subject>** contiene una breve descrizione della modifica seguendo queste linee guida:

- usare il modo imperativo: ad esempio, il presente "change", non "changed" né "changes";
- non usare la maiuscola per la prima lettera;
- nessun punto (.) alla fine;
- massimo 90 caratteri.

**<body>** è facoltativo e può essere utilizzato per descrivere i dettagli di questa modifica.

**<footer>** deve contenere:

- iniziare con "BREAKING CHANGE" se le modifiche 'rompono' l'API;
- riferimento al problema di GitHub o alla Pull Request, se applicabile. (Per i dettagli, vedere [Linking a pull request to an issue](#).)

Alcuni esempi:

```
fix(image): update size when a new source is set
```

```
fix(bar): fix memory leak
The animations weren't deleted in the destructor.
Fixes: #1234
```

```
feat: add span widget
The span widget allows mixing different font sizes, colors and styles.
It's similar to HTML <span>
```

```
docs(porting): fix typo
```

## 12.4 Developer Certification of Origin (DCO)

### 12.4.1 Panoramica

Per garantire che tutti i criteri di licenza siano soddisfatti per ogni repository del progetto LVGL, applichiamo un processo chiamato DCO (Developer's Certificate of Origin [Certificato di Origine dello Sviluppatore]).

Il testo del DCO è disponibile qui: <https://developercertificate.org/>.

Contribuendo a qualsiasi repository del progetto LVGL, si accetta che il contributo sia conforme al DCO.

Se il contributo soddisfa i requisiti del DCO, non sono necessarie ulteriori azioni. In caso di dubbi, non esitare a contattarci in un commento, ad esempio nella [Pull Request](#) inviata.

### 12.4.2 Licenze Accettate e Note di Copyright

Per rendere il DCO più facile da comprendere, ecco alcune guide pratiche su casi specifici:

## Il proprio lavoro

Il caso più semplice è quando il contributo è esclusivamente frutto del proprio lavoro. In questo caso si può semplicemente inviare una Pull Request senza preoccuparsi di problemi di licenza.

### Utilizzo di codice da un sorgente online

Se il codice da aggiungere è basato su un articolo, un post o un commento su un sito web (ad esempio StackOverflow), è necessario seguire la licenza e/o le regole di quel sito.

Ad esempio, nel caso di StackOverflow, è possibile utilizzare una nota come questa:

```
/* The original version of this code-snippet was published on StackOverflow.
 * Post: http://stackoverflow.com/questions/12345
 * Author: http://stackoverflow.com/users/12345/username
 * The following parts of the snippet were changed:
 * - Check this or that
 * - Optimize performance here and there
 */
... code snippet here ...
```

### Utilizzo di codice con licenza MIT

Poiché LVGL è con licenza MIT, altro codice con licenza MIT può essere integrato senza problemi. La licenza MIT richiede l'aggiunta di una nota di copyright all'opera derivata. Qualsiasi opera derivata basata su codice con licenza MIT deve copiare il file di licenza o il testo dell'opera originale.

### Utilizzare codice con licenza GPL

La licenza GPL non è compatibile con la licenza MIT. Pertanto, LVGL non può accettare codice con licenza GPL.

## 12.5 Stile di Codifica

### 12.5.1 File Template

Usare `misc/lv_tmpl.c` e `misc/lv_tmpl.h`

### 12.5.2 Convenzioni sui Nomi

- Le parole sono separate da `' '`
- Nei nomi di variabili e di funzioni utilizzare solo lettere minuscole (ad esempio `height_tmp`)
- Negli enum e nelle definizioni utilizzare solo lettere maiuscole (ad esempio `MAX_LINE_NUM`)
- Nomi globali (API):
  - iniziano con `lv`
  - seguiti dal nome del modulo: `button`, `label`, `style` ecc.
  - seguiti dall'azione (per le funzioni): `set`, `get`, ecc.
  - chiuso col soggetto: `name`, `size`, `state` ecc.
- Typedef
  - preferire `typedef struct` e `typedef enum` invece di `struct name` e `enum name`
  - `typedef struct` e `typedef enum` terminano sempre con `_t`
- Abbreviazioni:
  - Sono utilizzate e consentite le seguenti abbreviazioni:
    - \* `dsc` descrittore
    - \* `param` parametro
    - \* `indev` device di input

- \* **anim** animazione
  - \* **buf** buffer
  - \* **str** stringa
  - \* **min/max** minimo/massimo
  - \* **alloc** allocazione
  - \* **ctrl** control
  - \* **pos** posizione
- Evitare di aggiungere nuove abbreviazioni

### 12.5.3 Guida alla Codifica

- Editor:
  - Impostare l'editor in modo che utilizzi 4 spazi per le rientranze di tabulazione (invece dei caratteri di tabulazione).
  - Eccezione: il file **Kconfig** e tutti i file make richiedono caratteri di tabulazione iniziali sugli elementi figli.
- Funzioni:
  - Scrivere funzioni che utilizzano il principio di singola responsabilità.
  - Rendere le funzioni **static** quando non fanno parte dell'API pubblica di quell'oggetto (ove possibile).
- Variabili:
  - Una riga, una dichiarazione (BAD: char x, y;).
  - Usare **<stdint.h>** (*uint8\_t*, *int32\_t* ecc).
  - Dichiarare le variabili dove necessario (non tutte all'inizio della funzione).
  - Utilizzare lo scope più piccolo richiesto.
  - Le variabili in un file (al di fuori delle funzioni) sono sempre *static*.
  - Non utilizzare variabili globali (utilizzare funzioni per impostare/ottenere variabili statiche).

### 12.5.4 Commenti

Prima di ogni prototipo di funzione nei file .h, includere un commento formattato in Doxygen come questo:

```
/**  
 * Brief description. Add a blank line + additional paragraphs when more detail is needed.  
 * @param parent brief description of argument. Additional detail can appear  
 * on subsequent lines. List of accepted values:  
 * - value one  
 * - value two  
 * - value three  
 * @return brief description of return value.  
 */  
type_name_t * lv_function_name(lv_obj_t * parent);
```

Il normale prefisso di commento **/\*\*** fa sì che il commento documenti il pezzo di codice *dopo* il commento. Quando si documenta un pezzo di codice che si trova *prima* del commento, come un membro di una struttura, usare **/\*<** in questo modo:

```
/**  
 * Brief description of struct  
 *  
 * When more detail is needed, add a blank line then the detail.  
 */  
typedef struct {  
    char     *text;   /*< Brief description of this member */  
    uint16_t length; /*< Brief description of this member */  
} lv_example_type_t;
```

- Quando si commenta il codice, usare commenti a blocchi come questo **/\* Description \*/**, non commenti di fine riga come questo **// Description**.

- Includi uno spazio dopo /\* o /\*\*< e prima di \*/ per migliorare la leggibilità.
- Scrivere codice leggibile per evitare commenti descrittivi come: `x++; /* Add 1 to x */`.
- Il codice dovrebbe mostrare chiaramente cosa si sta facendo.
- Si dovrebbe scrivere il perché è stato fatto: `x++; /* Point to closing '\0' of string */`
- Sono accettati brevi "riassunti" di poche righe: `/* Calculate new coordinates */`
- Nei commenti, utilizzare le virgolette inverse (...) quando si fa riferimento a un elemento del codice, come una variabile, un tipo o il nome di una struttura: `/* Update value of `x_act` */`
- Quando si aggiungono o modificano commenti, le priorità sono (in ordine di importanza):
  1. chiarezza (la facilità con cui altri programmatore possono comprendere le vostre intenzioni),
  2. leggibilità (la facilità con cui altri programmatore possono leggere i vostri commenti),
  3. brevità (la qualità di usare poche parole quando si parla o si scrive).
- Le righe vuote all'interno dei commenti sono auspicabili quando migliorano la chiarezza e la leggibilità.
- Da ricordare, quando si scrive codice sorgente, non si sta solo insegnando al computer cosa fare, ma anche insegnando ad altri programmatore come funziona, non solo agli utenti dell'API, ma anche ai futuri manutentori del codice sorgente. I commenti aggiungono informazioni su cosa si stava pensando quando è stato scritto il codice e perché sono state fatte le cose in quel modo, informazioni che non possono essere trasmesse dal solo codice sorgente.

## Specifiche dei Commenti di Doxygen

Doxygen è il primo programma di una catena che genera la documentazione online dell'API LVGL dai file nel repository LVGL. Doxygen rileva i file a cui prestare attenzione tramite il comando `@file` all'interno di un commento Doxygen. I commenti Doxygen iniziano con /\*\*. Ignora i commenti che non contengono esattamente due \*.

Di seguito è riportata l'illustrazione di un prototipo di funzione API con documentazione che illustra la maggior parte dei comandi Doxygen utilizzati in LVGL.

```
/**  
 * Set alignment of Widgets placed in containers with LV_STYLE_FLEX_FLOW style.  
 *  
 * The values for the '..._place' arguments come from the `lv_flex_align_t`  
 * enumeration and have the same meanings as they do for flex containers in CSS.  
 * @param obj  
 *   pointer to flex container. It must have  
 *   'LV_STYLE_FLEX_FLOW' style or nothing will happen.  
 * @param main_place  
 *   where to place items on main axis (in their track).  
 *   (Any value of `lv_flex_align_t`.)  
 * @param cross_place  
 *   where to place item in track on cross axis.  
 *   - 'LV_FLEX_ALIGN_START'  
 *   - 'LV_FLEX_ALIGN_END'  
 *   - 'LV_FLEX_ALIGN_CENTER'  
 * @param track_cross_place  
 *   where to place tracks in cross direction.  
 *   (Any value of `lv_flex_align_t`.)  
 * Example for a title bar layout:  
 * @code{.c}  
 *   lv_obj_set_flex_align(title_bar, LV_FLEX_ALIGN_START, LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_CENTER);  
 * @endcode  
 * @see  
 *   - https://css-tricks.com/snippets/css/a-guide-to-flexbox/  
 *   - see `lv_obj_set_flex_grow()` for additional information.  
 */  
void lv_obj_set_flex_align(lv_obj_t * widget, lv_flex_align_t main_place, lv_flex_align_t cross_place,  
                           lv_flex_align_t track_cross_place);
```

- Iniziare sempre un commento Doxygen con una breve descrizione dell'elemento di codice che documenta.
- Quando sono necessari maggiori dettagli, aggiungere una riga vuota sotto la breve descrizione e aggiungere ulteriori informazioni che potrebbero essere necessarie agli utenti dell'API LVGL, incluse le precondizioni per la chiamata della funzione. Doxygen necessita della riga vuota per separare le parti "brevi" da quelle di "dettaglio" della descrizione.
- Descrivere i parametri della funzione col comando `@param`. Quando una funzione scrive all'indirizzo contenuto da un parametro puntatore, se non è già ovvio (ad esempio, quando il nome del parametro contiene la parola "out"), includere la direzione nel comando per chiarezza:

`@param[out] param_name descrizione.`

- Descrivere i valori restituiti col comando `@return`.
- Aggiungere almeno 2 spazi dopo i comandi Doxygen per una migliore leggibilità.

- Utilizzare le virgolette inverse ('...') attorno agli elementi del codice (variabili, nomi di tipo, nomi di funzione). Per i nomi di tipo e di funzione, Doxygen genera un collegamento ipertestuale alla documentazione di quel membro del codice (quando esiste), con o senza le virgolette inverse singole.
- Aggiungere un "()" vuoto ai nomi di funzione. Doxygen non genererà un collegamento ipertestuale alla documentazione della funzione senza questo.
- Utilizzare una grammatica corretta per chiarezza. Le descrizioni dei parametri non necessitano di punti dopo di esse, a meno che non siano frasi complete.
- Allineare i bordi del testo attorno agli elenchi di parametri per facilitarne la lettura.
- Gli elenchi (ad esempio, dei valori dei parametri accettati) possono essere creati utilizzando il carattere '-'. Se l'elenco deve essere numerato, è possibile utilizzare anche i numeri.
- Inserire il codice di esempio in un blocco di codice circondandolo con i comandi `@code{ . c }` e `@endcode`. Doxygen non necessita della parte `{ . c }`, ma il software a valle sì.
- Indirizzare il lettore a informazioni aggiuntive utilizzando il comando `@see`. Doxygen aggiunge un paragrafo "See also" [Vedi anche]. Il testo che segue il comando `@see` verrà indentato.
- Se si crea una nuova coppia di file `.c` e `.h` (ad esempio per un nuovo driver), assicurarsi che un commento Doxygen come questo sia presente all'inizio di ogni nuovo file. Doxygen non analizzerà il file senza il comando `@file`.

```
/**  
 * @file filename.c  
 *  
 */
```

### Comandi Doxygen Supportati

- `@file` indica a Doxygen di analizzare questo file e, se applicabile, fornisce anche la documentazione sul file stesso (tutto ciò che lo segue nello stesso commento).
- `@param name description` documenta `name` come parametro di funzione, e `description` è il testo che lo segue finché Doxygen non incontra una riga vuota o un altro comando Doxygen.
- `@return description` documenta il valore di ritorno finché Doxygen non incontra una riga vuota o un altro comando Doxygen.
- `@code{ . c }/@endcode` racchiude il codice che deve essere inserito in un blocco di codice. Mentre Doxygen sa utilizzare la codifica a colori C per i blocchi di codice in un file .C, la parte a valle della sequenza di generazione della documentazione non lo fa, quindi l'aggiunta di `{ . c }` al comando `@code` è necessaria.
- `@note text` inizia un paragrafo in cui è possibile inserire una nota. La nota termina con una riga vuota, la fine del commento o un altro comando Doxygen che inizia una nuova sezione. Se la nota contiene più di un paragrafo, è possibile aggiungerne altri utilizzando altri comandi `@note`. Al momento della stesura di questo documento, i comandi `@par` non aggiungono paragrafi aggiuntivi alle note, come indicato nella documentazione di Doxygen.
- `@see text` genera un paragrafo "See also" [Vedi anche] in una sezione evidenziata, utile quando si trovano altre prove delle informazioni aggiuntive su un argomento.

### 12.5.5 Convenzioni API

Per supportare la generazione automatica dei binding, l'API C LVGL deve seguire alcune convenzioni di codifica:

- Utilizzare `enum` invece delle macro. Se è inevitabile l'uso dei `define` esportarli con `LV_EXPORT_CONST_INT(define_value)` subito dopo il `define`.
- Negli argomenti delle funzioni, utilizzare la dichiarazione `type name[]` per i parametri dell'array invece di `type * name`
- Utilizzare puntatori tipizzati invece di puntatori `void*`
- Il costruttore del Widget deve seguire il pattern `lv_<widget_name>_create(lv_obj_t * parent)`.

- Le funzioni membro del Widget devono iniziare con `lv_<widget_name>` e devono ricevere `lv_obj_t*` come primo argomento, che è un puntatore all'oggetto Widget stesso.
- Le API `struct` devono seguire le convenzioni dei widget. Questo significa ricevere un puntatore alla `struct` come primo argomento, e il prefisso del nome della `struct` dovrebbe essere usato anche come prefisso del nome della funzione (ad esempio `lv_display_set_default(lv_display_t * disp)`)
- Le funzioni e le `struct` che non fanno parte dell'API pubblica devono iniziare con un carattere di sottolineatura per essere contrassegnate come "private".
- Anche l'argomento deve essere nominato nei file H.
- Non usare `malloc` in una variabile statica o globale. Invece, dichiarare la variabile nella struttura `lv_global_t` in `lv_global.h` e contrassegnarla con `(LV_GLOBAL_DEFAULT() ->variable)` quando viene utilizzata.
- Per registrare e utilizzare le callback, è necessario seguire una delle seguenti procedure.
  - Passare un puntatore a una `struct` come primo argomento sia della funzione di registrazione che del callback. Tale `struct` deve contenere il campo `void * user_data`.
  - L'ultimo argomento della funzione di registrazione deve essere `void * user_data` e lo stesso `user_data` deve essere passato come ultimo argomento della callback.

Per ulteriori informazioni, consultare la documentazione di [MicroPython](#).

## 12.5.6 Formattazione

Ecco un esempio per mostrare il posizionamento delle parentesi e l'uso degli spazi:

```
/**  
 * Set new text for a label. Memory will be allocated by label to store text.  
 *  
 * @param label pointer to label Widget  
 * @param text '\0' terminated character string.  
 *           NULL to refresh with current text.  
 */  
void lv_label_set_text(lv_obj_t * label, const char * text)  
{ /* Main brackets of functions in new line */  
  
    if(label == NULL) return; /* No bracket only if command is inline with if statement */  
  
    lv_obj_inv(label);  
  
    lv_label_ext_t * ext = lv_obj_get_ext(label);  
  
    /* Comment before a section */  
    if(text == ext->txt || text == NULL) { /* Bracket of statements starts on same line */  
        lv_label_refr_text(label);  
        return;  
    }  
  
    ...  
}
```

È possibile utilizzare `astyle` per formattare il codice. Eseguire `code-format.py` dalla cartella `scripts`.

## 12.5.7 Include

Vari sottosistemi di LVGL possono essere abilitati o disabilitati impostando una macro in `lv_conf.h` rispettivamente su 1 o 0. I file di codice che contengono la logica per tali sottosistemi sono spesso organizzati in modo che vi sia una direttiva `#if <ENABLING_MACRO_NAME>` all'inizio del file e la sua corrispondente `#endif` alla fine. Se si aggiunge o si modifica un sottosistema di questo tipo in LVGL, ove possibile, l'unica `#include` che dovrebbe trovarsi sopra tali direttive condizionali dovrebbe essere sufficiente a includere la macro di abilitazione/disabilitazione. In particolare:

- nel file `.c`: l'`#include` che include l'header con la corrispondenza più vicina a quel file `.c`
- nel file `.h` corrispondente: `#include "lv_conf_internal.h"`

che, a sua volta, include `lv_conf.h`. Vedere gli esempi in `lv_freetype.c`, `lv_freetype_private.h` e `lv_freetype.h`.

## 12.5.8 pre-commit

pre-commit è un gestore di pacchetti multilinguaggio per gli hook di pre-commit. Consultare la [installation guide](#) per installare il pacchetto Python di pre-commit sul computer di sviluppo.

Una volta installato `pre-commit` si dovranno configurare gli script degli hook di git con:

```
pre-commit install
```

ora `pre-commit` verrà eseguito automaticamente su `git commit`!

## 12.5.9 Hook

L'hook locale `format-source` (vedere `.pre-commit-config.yaml`) esegue `astyle` su tutti i file sorgente e header in staging (che non sono esclusi, vedere la chiave `exclude` della configurazione dell'hook) prima di inserire il messaggio di commit. Se un file viene formattato da `astyle` sarà necessario aggiungere la modifica all'area di staging ed eseguire nuovamente `git commit`.

L'hook `trailing whitespace` corregge gli spazi vuoti finali su tutti i file.

## 12.5.10 Saltare gli hook

Si può saltare un hook in particolare, con:

```
SKIP=name-of-the-hook git commit
```

## 12.5.11 Hook di test

Non è necessario eseguire un commit per testare gli hook, si possono testare aggiungendo i file nell'area di staging ed eseguendo:

```
pre-commit run name-of-the-hook
```

# CAPITOLO 13

---

## Change Log

---

### 13.1 v9.3.0 3 June 2025

#### 13.1.1 Summary

This is a significant release packed with exciting updates from both our community and internal team.

One of the most notable additions is **XML support**, enabling you to describe your UI in a declarative manner. This way LVGL UI can be loaded at runtime without recompiling and deploying the application. This feature is also a key requirement for the upcoming **UI editor** we're developing.

The [documentation](#) has been thoroughly reorganized and proofread.

We've also laid the groundwork to support **3D textures** as LVGL widgets.

To support our partners, we've added numerous new drivers and documentation:

- STM's DMA2D
- STM's NeoChrom GPU
- STM's LTDC LCD peripheral
- Generic NemaGFX driver
- NXP's G2D GPU
- UEFI BIOS driver
- Toradex documentation
- Torizon OS guide
- Buildroot integration
- Added dmabuf support to the [Wayland driver](#) (currently limited to G2D draw unit)
- Hotplug support in evdev with auto-discovery
- Optional use of [Linux CPU usage statistics](#)

**Text rendering** has become much more advanced:

- Our [FreeType binding](#) now supports colored glyphs and glyph outlines
- Text recoloring is back (e.g., "A #ff0000 red# word")

- Improved GPU integration to render A1/A2/A4 bitmaps directly

And finally, some other important additions:

- New color formats: `RGB565_SWAPPED` (commonly used with SPI-based display controllers) and `ARGB8888_REMULTIPLIED` (required for [Wayland](#) and [Lottie](#) animations)
- [SVG support](#)
- [Double and triple click detection](#)
- Tiled rendering to better utilize multiple CPU cores
- [Multi-touch gestures](#) (swipe, pinch, rotate)
- [Global recolor style property](#) (tint all widgets and images)
- Triple buffer support

### 13.1.2 Architectural

- `arch(font_manager): add multiple font backend support` 8038
- `arch(demos): move new demos to lv_demos` 8022
- `arch(cmake): add native Kconfig support to cmake` 7934

### 13.1.3 New Features

- `feat(wayland): add dmabuf support with g2d` 8122
- `feat(draw_sw): add RGB565_SWAPPED support` 8227
- `feat: add new lvgl examples` 8225
- `feat(barcode): support raw code 128` 8287
- `feat(draw_opengles): add getter fn for glfw_window and fix freeing non allocated textures` 8257
- `feat(draw): add configurable thread priority for all drawing units` 8162
- `feat(test): add Dockerfile support with CI env` 8209
- `feat(image): support symbol images with inner alignment` 8182
- `feat(disp): allow rotation with `FULL` render mode` 8107
- `feat(display): add triple buffer support` 8158
- `feat(svg): add API for getting original width and height` 8180
- `feat(cache): add new `lv_cache_lru_ll` module` 8155
- `feat(cmake): disable PCPP by default` 8126
- `feat(animimage): support set source interfaces with or without parameter of reverse play` 8164
- `feat(drivers): add ft81x framebuffer driver` 7815
- `feat(animimage): support images play in a reversed order` 8085
- `feat(font): Replace SimSun font with SourceHanSansSC` 8006
- `feat(widget): Add `lv_3dtexture` widget and 3D draw task type` 8033
- `feat(nuttx): auto enable display matrix rotation` 8119
- `feat(refr): use transform matrix to realize display rotation` 6911
- `feat(image): add LV_IMAGE_ALIGN_CONTAIN & LV_IMAGE_ALIGN_COVER to scale images without changing aspect ratio` 7955
- `feat(sdl): add window icon settings` 7808
- `feat(sw_blend): add argb8888-premultiplied support` 7979
- `feat(linux/fbdev): support non-mmappable frame buffers` 8058
- `feat(disp): support subscription and unsubscription of vsync event` 7487
- `feat(demos): add WiFi credentials input to the high res demo` 7953
- `feat(scale): add id1 id2 for tick line draw descriptors. fix zero division` 7695
- `feat/scripts): Add --name parameter to LVGLImage.py` 7996

- **feat(draw/sw): add support for vector fonts** [7560](#)
- **feat(draw): add comments and unify struct names** [7878](#)
- **feat(refr): add global recolor** [7855](#)
- **feat(lz4): update lz4 to 0.10.0** [7869](#)
- **feat(test): add test cases for indev\_gesture pinch** [7947](#)
- **feat(chart): implement get\_index\_from\_x() for LV\_CHART\_TYPE\_SCATTER** [7824](#)
- **feat(demo): New smartwatch demo** [7883](#)
- **feat(indev): Add rotation and two fingers swipe gestures support** [7865](#)
- **feat(obj\_tree): indent printed object data according to its depth** [7852](#)
- **feat(draw\_sw): have only one SW draw unit with multiple threads internally** [7899](#)
- **feat(dma2d): add support for DMA2D on STM32H7RS** [7850](#)
- **feat(benchmark): add on benchmark end callback** [7814](#)
- **feat(tests): add draw 8bpp font test** [7895](#)
- **feat(spangroup): add some testcases for span rtl mode.** [7870](#)
- **feat(examples): add looping scroll example** [7714](#)
- **feat(draw/sw): allow custom handlers** [7531](#)
- **feat(nxp): add G2D support** [7625](#)
- **feat(lv\_bin\_decoder): improve error logging in file reading** [7826](#)
- **feat(blend): add blend difference mode** [7796](#)
- **feat(thorvg): use LVGL's malloc/realloc/zalloc/free** [7772](#)
- **feat(nuttx\_image\_cache): add configuration to use image cache heap for default heap** [7653](#)
- **feat(vg\_lite): optimize resource reference count management** [7809](#)
- **feat(mem): add lv\_reallocf** [7780](#)
- **feat(span): support bidi for span** [7717](#)
- **feat(draw\_vector): add fill units support** [7774](#)
- **feat(docs): new docs-build paradigm...** [7597](#)
- **feat(docs): upgrade presentation of display.rst** [7478](#)
- **feat(decoder): add svg image decoder for image widget** [7141](#)
- **feat(osal): add lv\_os\_get\_idle\_percent for linux** [7632](#)
- **feat(fs): clarify rest of docs on driver-identifier letters.** [7710](#)
- **feat(vg\_lite): enhanced the automatic diagnosis function of GPU errors** [7751](#)
- **feat(demos): add new card to high resolution demo** [7699](#)
- **feat(observer): add bind\_XXX ge/gt/le/lt and notify only when value changes** [7678](#)
- **feat(drm): add support for GBM buffer object to increase performance** [7464](#)
- **feat(draw/sw): make I1 luminance threshold configurable** [7616](#)
- **feat(drivers): add evdev discovery** [7481](#)
- **feat(docs): proofread/edit obj\_property.rst.** [7621](#)
- **feat(demo): add smartwatch demo** [7429](#)
- **feat(scripts/LVGLImage.py): adds RGB565 dithering support** [7582](#)

- **feat(draw): add lv\_draw\_letter support** [7490](#)
- **feat(anim): add a pause method** [7583](#)
- **feat(docs): batch 14 of proofread/edited docs** [7477](#)
- **feat(docs): add short bit about include pattern to CODING\_STYLE.rst** [7563](#)
- **feat(docs): batch 12 of proofread/edited docs** [7440](#)
- **feat(dma2d): add support for ARGB1555 color on top of RGB565 format** [7555](#)
- **feat(drivers): add UEFI driver** [7069](#)
- **feat(demos): High Resolution Demo Improvements** [7566](#)
- **feat(xml): add support for more properties and add more examples** [7417](#)
- **feat(libinput): map LV\_KEY\_{ESC}** [7544](#)
- **feat(docs): batch 11 of proofread/edited docs** [7361](#)
- **feat(docs): batch 10 of proofread/edited docs** [7357](#)
- **feat(docs): batch 13 of proofread docs** [7458](#)
- **feat.dropdown): add lv\_anim\_enable\_t parameter to lv\_dropdown\_set\_selected** [7310](#)
- **feat(demos): add high resolution demo** [7308](#)
- **feat(draw/sw): added support for 3 bpp font rendering** [7350](#)
- **feat(gen\_json): adds option to gen\_json to build without docstrings** [7471](#)
- **feat(docs): make docs development practical** [7414](#)
- **feat(osal): add SDL2 based threading and OS support** [7457](#)
- **feat(anim): clarify reverse play in animation API** [7338](#)
- **feat(draw): add layer memory allocation config support** [7038](#)
- **feat(NemaGFX): add freetype vector font support** [7346](#)
- **feat(refr): improve performance measurement** [7430](#)
- **feat(display): add draw buffer size getter** [7332](#)
- **feat(docs): proofread/edit batch 9** [7324](#)
- **feat(docs): widget proofread wrap-up** [7405](#)
- **feat(docs): update threading details in answer to #7313** [7342](#)
- **feat(profiler\_builtin): support nanosecond accuracy** [7415](#)
- **feat(vg\_lite): reduce unnecessary path quality settings** [7398](#)
- **feat(examples): add infinite scroll example** [7388](#)
- **feat(docs): batch 8 of proofread/edited docs** [7295](#)
- **feat(docs): document LV\_DPX(n)** [7374](#)
- **feat(file\_explorer): remove '.' and rename '..' to '&lt; Back'** [7270](#)
- **feat(sdl): add I1 color format render support** [7036](#)
- **feat(drivers/st\_ltdc): add rotation support to LTDC driver** [7254](#)
- **feat(chart,calendar): two grammar corrections** [7340](#)
- **feat(scroll): user-defined scrollbar length using LV\_STYLE\_LENGTH** [7306](#)
- **feat(draw\_sw): add image clip\_radius and mask before transformation** [7244](#)
- **feat(doc): add documentation on Torizon OS** [7280](#)

- **feat(docs): batch 6 of proofread/edited docs** [7277](#)
- **feat(scroll): adjust scroll behavior for non-elastic objects** [7336](#)
- **feat(code-format): minor enhancements** [7311](#)
- **feat(font): allow using A1,2,4 bitmaps + handle byte aligned fonts** [7234](#)
- **feat(fsdrv): set the working directory** [7272](#)
- **feat(observer): add subject snprintf** [7250](#)
- **feat(indev): add multi touch gestures** [7078](#)
- **feat(docs): batch 7 of proofread/edited docs** [7281](#)
- **feat(png): add support for files without extension** [7289](#)
- **feat(docs): scrolling doc proofread, edited and clarified...** [7170](#)
- **feat(issue): add platform description to bug-report** [7273](#)
- **feat.dropdown): add animations on rotary event** [7271](#)
- **feat(docs): batch 5 of proofread/edited docs** [7218](#)
- **feat(indev): add setter for long press repeat time** [7235](#)
- **feat(docs): buttonmatrix proofread/edit** [7194](#)
- **feat(obj): add transform matrix attribute** [7187](#)
- **feat(sdl): use SDL\_Delay for delay callback** [7243](#)
- **feat(docs): batch 4 of proofread/edited docs** [7207](#)
- **feat(observer): add null pointer check** [7183](#)
- **feat(docs): plea for proper word-wrapping in `rst` files.** [7189](#)
- **feat(thorvg): update thorvg to 0.15.3** [7103](#)
- **feat(docs): batch 3 of proofread/edited docs** [7180](#)
- **feat(sw): Add method to convert a htiled I1 buffer to vtiled** [7129](#)
- **feat(fs\_posix): add error code conversion** [7166](#)
- **feat(roller): set roller option with a string** [7143](#)
- **feat(docs): proofread and edited docs** [7144](#)
- **feat(draw\_label): Support simultaneous text selection and recolor** [7116](#)
- **feat(font): support 8 bpp font bitmaps** [7100](#)
- **feat(vg\_lite): add more detailed error dump information** [7104](#)
- **feat(ffmpeg): add playback complete event trigger** [7119](#)
- **feat(docs): reorganize docs** [7136](#)
- **feat(scale): add additional style properties** [6649](#)
- **feat(vg\_lite): add ARGB1555 ARGB4444 ARGB2222 support** [7028](#)
- **feat(sdl): add float zoom window support** [7089](#)
- **feat(scripts/gdb): add draw unit debug info support** [7095](#)
- **feat(tests): add log print callback** [7076](#)
- **feat(nuttx): add indev cursor display** [7021](#)
- **feat(libs): add SVG rendering support** [6845](#)
- **feat(drivers): add STM32 LTDC support** [7059](#)

- **feat(demo): add an ebike demo** [7019](#)
- **feat(opengl): texture caching** [6861](#)
- **feat(draw): add NemaGFX rendering backend** [7002](#)
- **feat(text): add text recolor back** [6966](#)
- **feat(vg\_lite): add profiler for vg\_lite\_set\_scissor** [7023](#)
- **feat(vg\_lite\_tvg): add vg\_lite\_set\_scissor function support** [6959](#)
- **feat(nxp): Release/nxp patches for LVGL master** [6978](#)
- **feat(render): basic of tiled rendering** [6761](#)
- **feat(sdl): speed up rotation** [6835](#)
- **feat(nuttx): move static var index to global for gdb diagnostic** [6890](#)
- **feat(style): add a "set all" function for margin** [6904](#)
- **feat(gif): add loop count control** [6839](#)
- **feat(snapshot): add argb8565 support** [6899](#)
- **feat(array): add an empty element when element is NULL** [6893](#)
- **feat(vg\_lite): add A8 and L8 dest buffer support** [6884](#)
- **feat(animimg): add getter function for underlying animation** [6838](#)
- **feat(vg\_lite): add yuy2 color format support** [6882](#)
- **feat(profiler): support different module divisions** [6834](#)
- **feat(switch): add vertical switch function** [6786](#)
- **feat(spangroup): add `lv\_spangroup\_get\_span\_by\_point` in spangroup** [6579](#)
- **feat(indev): detect double and triple click (closes #6020)** [6187](#)
- **feat(dma2d): add basic support** [6691](#)
- **feat(nuttx): add memory dump for image cache heap** [6807](#)
- **feat(vg\_lite): optimize check\_image\_is\_supported** [6802](#)
- **feat(vg\_lite): add draw border side support** [6796](#)
- **feat(freetype): add colored glyphs support** [6686](#)
- **feat(event): event supports delayed deletion ability** [6655](#)
- **feat(libs): add freetype font manager** [6482](#)
- **feat(stdlib): add lv\_calloc function** [6743](#)
- **feat(fonts) add aligned fonts for demo benchmark** [157ee6a](#)
- **feat(xml): add the basics of declarative XML support** [fc5939d](#)
- **feat(test): make LVGL's test utilities public** [8d04466](#)
- **feat(xml): add support scale, span, roller and bar** [a535063](#)
- **feat(circle\_buf): add lv\_circle\_buf\_t component** [d08d545](#)
- **feat(xml): add subject and global/local scoping support** [2ca425c](#)
- **feat(xml): add buttonmatrix** [299e31b](#)
- **feat(obj\_name): add auto-indexing with names like 'mybtn\_#'** [e4bbc4f](#)
- **feat(xml): add gradient support** [b78a9b4](#)
- **feat(gdb): add lvgl GDB plugin** [d460edb](#)

- **feat(xml): add basic callback event support** [1c9c9f6](#)
- **feat(xml): add canvas and calendar support** [79381cc](#)
- **feat(xml): add support for textarea and keyboard** [b63472d](#)
- **feat(xml): add arc and checkbox** [3095b63](#)
- **feat(xml): add obj, roller, dropdown, and arc bind\_\* properties** [82996a5](#)
- **feat(iter): add lv\_iter\_t module** [4d7f577](#)
- **feat(xml): load fonts and images from XML** [670845e](#)
- **feat(xml): add test for a complex view** [052d908](#)
- **feat(xml): add the first version of property updater** [b23a228](#)
- **feat(image\_cache): add dump info ability for image\_cache** [30f0d6c](#)
- **feat(property): support second value in property** [133b6fc](#)
- **feat(xml): support removing style properties** [ed60b20](#)
- **feat(freetype): add font kerning support** [0414c78](#)
- **feat(slider): add property interface** [4eacf15](#)
- **feat(vglite) draw\_vglite\_border add support for legacy scissor** [2e75bbd](#)
- **feat(animimage): add property support** [f5b00f5](#)
- **feat(draw\_sw): draw A8 static fonts directly in SW render** [102b633](#)
- **feat(test/freetype): add vg-lite outline font test** [feee9ae](#)
- **feat(xml): handle registering assets multiple times** [910cf08](#)
- **feat(NemaGFX): add TSC color formats** [07854b1](#)
- **feat(slider): add orientation support (based on bar)** [f23b42b](#)
- **feat(vglite) add LV\_USE\_VGLITE\_STATIC\_BITMAP option to use static glyphs for text rendering** [7ba6cf2](#)
- **feat(chart): add lv\_chart\_set\_cursor\_pos\_x/pos\_y()** [8d0a519](#)
- **feat(nuttx\_fbdev): support for complete non-/consecutive fbdev mmap processing** [340482d](#)
- **feat(label\_xml): support the format string in bind\_text** [c5022ce](#)
- **feat(xml): support styles with the same name** [7f45e2f](#)
- **feat(observer): support simple int binding to label** [c23636c](#)
- **feat(observer): rename arg to clarify correct use...** [82034cb](#)
- **feat(xml): use LVGL's malloc for parsing** [717301a](#)
- **feat(obj): add state processing in XML parser** [81eb192](#)
- **feat(xml): add support for ext\_click\_area** [d91bdc5](#)
- **feat(label\_xml): support the format string in bind\_text** [3d1beab](#)
- **feat(obj): add obj name support** [626d6b5](#)
- **feat(observer): Make `lv\_obj\_remove\_from\_subject()` legal for Widget Binding.** [9b3c365](#)
- **feat(lv\_subject\_t): re-sequence fields to reduce size from 32 => 28 bytes** [e0dfe80](#)
- **feat(indev): detect double and triple click (closes #6020) (#6187)** [651f69f](#)

### 13.1.4 Performance

- perf(blend): fix loop unrolling condition 8306
- perf(vg\_lite): relaxing the matrix transform judgment criteria 8219
- perf(label): reduce the time of calling lv\_text\_get\_size when drawing 8129
- perf(vg\_lite): switch gradient cache lru\_rb -> lru\_ll 8188
- perf(vg\_lite): add gradient pool to optimize memory allocation 8149
- perf(refr): reduce refr\_children call by check layer->opa 8133
- perf(draw): support draw task dsc alloc together to reduce the malloc call times 8070
- perf(draw): reduce empty dispatch 8073
- perf(vg\_lite): add font cache reference counting support 7407
- perf(array): make short functions inline to reduce function jumps 7890
- perf(vg\_lite): reduce ineffective D-cache flushing operations 7817
- perf(draw): optimize lv\_draw\_label() with text\_length for text\_local 7673
- perf(obj): use layer to cache the current opa stack 7523
- perf(nxp/vglite): do not recreate identity matrix at runtime 7595
- perf(vg\_lite): optimize matrix and rectangle conversion 7537
- perf(refr): reduce matrix conversion 7536
- perf(vg\_lite): improve path append data performance 7504
- perf(bin\_decoder): improve the decoding performance of a8 7494
- perf(vg\_lite): improve path data conversion performance 7470
- perf(os): optimize OS API calls without OS mode 7420
- perf(span): optimize span render performance up by 50% 7290
- perf(draw): improve execute time of obj's transforming 6638
- perf(vg\_lite): reduce redundant matrix calculations 7163
- perf(label): simplify handling of bytes overwritten with dots 7001
- perf(benchmark): use XRGB8888 image with 32bit color depth 6914
- perf(vg\_lite): optimize label drawing efficiency 6853
- perf(draw): skip area independence tests with one draw unit 6825
- perf(obj): skip repeated flag setting operations 6859
- perf(obj): reduce unnecessary border post drawing 6852
- perf(vg\_lite): reduce matrix and radius calculations 6800
- perf(vg\_lite): balancing performance and memory consumption 6823
- perf(demo): use set\_text\_static in demos to cache them in SDL and OpenGL renderers 926e3df

### 13.1.5 Fixes

- fix(nema\_gfx): fix STM32U5 compilation with NEMAGFX enabled and NEMAVG disabled 8291
- fix(vg\_lite): add nullptr check for vg\_dash\_pattern 8352
- fix(example): fix params to lv\_rand call 8324
- fix(NemaGFX): get the static bitmap even if bpp is not 8 8347

- fix(NemaGFX): new label static bitmap handling and fix area move bug 8332
- fix(esp/nuttx): add missing include 8344
- fix(display): fix divide by stride of zero value 8308
- fix(calendar): allow setting years in ascending order 8315
- fix(calendar): fixed macro comparing unsigned value to 0 8316
- fix(sysmon): define perf monitor log mode when perf monitor is not enabled 8329
- fix(themes): Fix mono theme init 8343
- fix(chart): fix variable overflow 8318
- fix(evdev): mark unused variable as unused 8330
- fix(span): fix dereference obj before checking for null 8317
- fix(draw\_img): pass correct variable to LV\_DRAW\_SW\_IMAGE 8314
- fix(draw\_sw): in ARGB8888\_PREMULTIPLIED fix rounding error and add RGB888 image blending support 8264
- fix(xml): fix tabview XML definition 8309
- fix(freetype\_image): dereference null pointer 8307
- fix(theme): fix judgment logic error. 8303
- fix(pxp): sync rotation direction with SW render 7063
- fix(draw\_vector): draw vector not calc draw\_buf offset 8262
- fix(svg): correctly parse header 8281
- fix(vg\_lite): fix strict alias warning with higher optimization levels 8136
- fix(tiny\_ttf): fix GPOS lookup list table address 8251
- fix(layout): size in cross direction was not updated when layout on parent was changed from grid to flex 8053
- fix(wayland): prevent wayland reinitialization to support multiple windows 8273
- fix(wayland): assert surface is configured after creating window 8271
- fix(bar): fix bar indicator length error 8200
- fix(vg\_lite): fix access to uninitialized members 8256
- fix(SW) build with LV\_DRAW\_SW\_COMPLEX disabled 8246
- fix(vg\_lite): enable box shadow by default 8242
- fix(dave2d): fix evaluate callback 8253
- fix: format specifier in lv\_obj\_tree.c 8244
- fix(refr): add missing area intersect check 8240
- fix(refr): fix matrix rotation precision loss 8221
- fix(bin\_decoder): fix build warnings when decompressing LZ4 compressed images 8234
- fix(vg\_lite): reset the scissor area when rendering with vg\_lite. 8232
- fix(pinyin): update candidates on the second letter too 8105
- fix(wayland): support version 2 of the XDG protocol 8201
- fix(log): fix possible level out of bounds 8216
- fix(nxp/g2d): remove useless g2d\_search\_buf\_map() when free buf 8072
- fix(wayland): Remove the XDG\_RUNTIME\_DIR check 8041

- fix(span): replace deprecated spangroup\_set\_mode fn from examples 8192
- fix(thorvg): fix incompatibility with c++20 (#8042) 8043
- fix(lottie): smooth edges on lottie animation 8189
- fix(roller): don't send click event when scrolled 8101
- fix(span): add assertion for bidi text buffer allocation 8168
- fix(display/renesas\_glcde): stride of the framebuffer can be different to the width 8177
- fix(dave2d): fix compilation error 8175
- fix(circle\_buf): don't clear array on reset 8157
- fix(refr): change clip area don't take effect on children and draw post 8117
- fix(examples): fix #warning causing examples build to fail. 8135
- fix(list): check that LV\_USE\_FLEX is enabled when using LV\_LIST 8139
- fix(opa): corrected incorrect usage of LV\_OPA\_MIN 8161
- fix(nuttx\_image\_cache): fix compilation issues 8165
- fix(NemaGFX): update STM32U5 nema lib to revC 8138
- fix(drivers/evdev): process pointer coordinates in unrotated frame 8061
- fix(vg\_lite): fix vector draw pattern matrix error 8134
- fix(arc): handle clicks on a full circle 8106
- fix(cache): prevent resource leaks in cache entry creation failure 8144
- fix(indev): fix indev gesture occasional crash 8146
- fix(ffmpeg): fix ffmpeg decoder assert 8128
- fix(makefile): Extend component.mk with missing elements 8118
- fix(tabview):fix example lv\_example\_tabview\_2 8084
- fix(vg\_lite): fix vector drawing not handling global matrix 8115
- fix: var redeclaration 8109
- fix(cmake): fix LV\_CONF\_PATH quoting consistency issue 8079
- fix(sdl): fix access fb2 NULL pointer 8096
- fix: table get cell area error 8047
- fix(test): do not enable test\_bg\_image with || 1 8081
- fix(makefile): Path fixes in component.mk 8068
- fix(area): test and fix lv\_area\_diff edge case 7907
- fix(font): allow non-constant LV\_FONT\_DEFAULT again (fixes #7788) 8059
- fix(draw asm): replace attribute syntax for asm arm files 8076
- fix(matrix): use homogeneous coordinates to transform point 7960
- fix(draw): fix "blend\_non\_normal\_pixel: Not supported blend mode" issue when using lv\_demo\_smartwatch which compiled by MSVC 8017
- fix(draw\_chart): remove raw\_end setting in draw\_series\_line 8024
- fix(draw\_sw): fix rotation typos 8050
- fix(lottie): revert example 8054
- fix(draw asm): fix GCC linker error 8055

- fix(tree): add NULL check 7972
- fix(draw\_sw): make the images invisible on full recolor to the background color 7868
- fix(text): handle recolor in lv\_text\_get\_size 8026
- fix(observer): check if observer is associated with obj on remove fn 7727
- fix(example\_anim): remove scrollable flag 8008
- fix(lv\_image): fix touch area calculation 8027
- fix(vg\_lite): check for better draw unit 8040
- fix(draw\_sw): fix letter outline multi-threading issues 8003
- fix(docs): fix Riverdi & Viewe links 8031
- fix(draw\_nema\_gfx\_triangle): use correct field names for lv\_draw\_triangle\_dsc\_t 8005
- fix(getcwd): capture return value #7991 7992
- fix(label): fix long mode clip #7922 7957
- fix(svg): adjust svg render node object for reduce memory usage. 8013
- fix(micropython): lvgl module\_deinit 7973
- fix(draw/sw): draw outline span wrong init. 8011
- fix(tests): explicitly install libudev dependency #7983 7985
- fix(scripts): initialize reserved fields of lv\_image\_dsc\_t and lv\_image\_header\_t to prevent compiler warnings 7799
- fix(demo): add LV\_USE\_LOTTIE check for demo smartwatch 8002
- fix(gif): free memory on error 7950
- fix(draw\_sw\_vector): fix thorvg canvas colorspace 7975
- fix(thorvg.h): ensure can use standard ints #7988 7989
- fix(color): add cast to LV\_OPA\_MIX macros 7956
- fix(Widgets): Optimise widget event callbacks to remove unnecessary calls to the event handler. 7954
- fix(vg\_lite): modify recolor image behavior 7977
- fix(wayland): Ensure variable is initialized before use #7986 7987
- fix(bin\_decoder): fix the crash when decoder A8 images and flush cache 7952
- fix(image\_decoder): enhance code robustness 7969
- fix(anim): fix the crash caused by delete anim in cb 7926
- fix(obj): remove all events from the object 7811
- fix(buttonmatrix): initialize auto\_free\_map in constructor 7966
- fix(draw\_sw): fix memory leaks 7964
- fix(nxp/g2d): fix memory management error in G2D buffer mapping 7875
- fix(thorvg): fix internal types of rasterXYFlip 7782
- fix(port\_releaser): always use remote when pushing and add token 7856
- fix(X11): call XCloseDisplay in delete 7904
- fix(refr): make lv\_display\_refr\_timer public again 7925
- fix(scale): remove dangling reference to stack buffer 7915
- fix(examples): add casts and change int types to fix checker warnings 7933

- fix(observer): fix lv\_subject\_notify\_if\_changed not defined when LV\_USE\_DROPDOWN = 0 [7783](#)
- fix(decoder): missing log\_trace when decoders are not found. [7790](#)
- fix(indev): remove redundant zero-initialization operations [7929](#)
- fix(scale): change return type of lv\_scale\_get\_rotation to int32\_t and update documentation [7862](#)
- fix(obj): style opa of other part is not effective [7905](#)
- fix(nuttx): incorrect draw buffer size for I1 color and bpp=1 [7885](#)
- fix(label): fix behavior when set to `LV\_LABEL\_LONG\_MODE\_SCROLL\_CIRCULAR` with BIDI [7886](#)
- fix(svg): fix SVG draw rect off by 1px bug [7902](#)
- fix(dave2d): LV to D2 colour format conversion for I8 format [7896](#)
- fix(dave2d): fix build break [7882](#)
- fix(thorvg): add missing include [7877](#)
- fix(nuttx): fix nuttx lcd release assert [7840](#)
- fix(refr): fix invalidate area calc error [7871](#)
- fix(vg\_lite): fix missed reference count release [7889](#)
- fix(freetype): add missing error handling for glyph bitmap lookup failure [7887](#)
- fix(vg\_lite): fix pending swap sequence error [7849](#)
- fix(indev): fix lv\_indev\_gesture write access error [7843](#)
- fix(SDL): error if color depth is 1 and render mode is not partial [7846](#)
- fix(draw): fix 'lv\_vector\_path\_append\_arc' not starting with a 'move to' operation. [7854](#)
- fix(release\_updater): push: if branch is specified, remote must be specified too [7863](#)
- fix(obj): cover check should consider both grad opa-s [7813](#)
- fix(indev): fix platform-specific printf format for int32\_t [7844](#)
- fix(profiler\_builtin): fix uint32\_t storage nanosecond time overflow [7818](#)
- fix(demos/smartwatch): fix image color format error [7819](#)
- fix(nema): Broken build after draw unit refactoring, draw\_label changes [7759](#)
- fix(release\_updater): minor fixes and addition of new boards [7726](#)
- fix(examples): add casts [7831](#)
- fix(indev): fix scroll\_obj not send LV\_EVENT\_INDEV\_RESET [7767](#)
- fix(kconfig): add LIBUV dependency to prevent missing configurations [7798](#)
- fix(indev): modify calculation delta\_y use p\_delta\_y, not p\_delta\_x [7791](#)
- fix(example\_grad): return value from lv\_style\_get\_prop not checked [7793](#)
- fix(indev): fix platform-specific printf format for int32\_t [7784](#)
- fix(draw\_letter): fix draw letter bg\_coords behaviour when it's NULL [7773](#)
- fix(text): fix oob read for utf8-next [7602](#)
- fix(bin\_decoder): check divisor is zero [7610](#)
- fix(nema): fix indexed image error [7744](#)
- fix(DMA2D): build issue after the draw\_unit to draw\_task refactor [7749](#)
- fix(vg\_lite): alleviate the loss of precision in obtaining path bound [7731](#)

- fix(indev): add the missing wait\_until\_release flag to clean up 7638
- fix(draw): fix wrong LV\_PROFILER tag 7737
- fix(test\_anim): fix stack-use-after-return 7730
- fix(flex): don't count item gap for leading hidden items 7720
- fix(drivers): generic MIPI add missing lv\_display\_flush\_ready 7693
- fix(file\_explorer): fix navigation when using a keypad indev 7181
- fix(area): lv\_area\_diff remove overlap 7696
- fix(indev.rst): handle lv\_obj\_is\_focused no longer exists. 7711
- fix(sysmon): disable all performance banners from screen with serial redirect 7593
- fix(draw\_sw\_mask): add ASSERT\_MALLOC check 7692
- fix(memcpy): ensure volatile qualifier for destination pointer in lv\_... 7573
- fix: warn user about deprecated LV\_DEFAULT\_DRIVE\_LETTER 7620
- fix(chart): fix last point not drawn in scatter chart 7665
- fix(canvas): initialize layer before return 7677
- fix(indev): skip press event on new object release 7612
- fix(xml): Fixed small typo 7684
- fix(docs): clarify obj hierarchy after `lv\_menu\_page\_create()` 7604
- fix(refr): lv\_obj\_invalidate\_area invalidates whole obj 7598
- fix(gif): fix bounds check 7675
- fix(docs): fix minor issues with alif.rst 7676
- fix(chart): avoid divide by zero 7609
- fix(docs): several minor doc fixes 7652
- fix(cmake): installation with custom LV\_CONF\_PATH 7624
- fix(fs\_win32): fix inconsistency using LV\_FS\_WIN32\_PATH... 7608
- fix(layout): always recalculate the layout if HIDDEN changes 7607
- fix(draw\_vector): alleviate the loss of precision in obtaining path bound 7635
- fix(refr): lv\_refr\_get\_top\_obj not check style opa 7643
- fix(vg\_lite\_math): initialize all variables to avoid error of uninitialized variables on some compilers. 7628
- fix(nuttx\_image\_cache): fix incorrect code order 7644
- fix(span): fix align text to center and right layout issues. 7615
- fix(vg\_lite): remove pattern color fill 7613
- fix(nxp): allow vglite build when LV\_USE\_DRAW\_SW is disabled 7596
- fix(wayland): use premultiplied alpha for transparent backgrounds #7543 7580
- fix(os): add support for thread names 7579
- fix(menu): add missing LV\_ASSERT\_OBJ 7605
- fix(dave2d): handle LV\_COLOR\_FORMAT\_RGB888 correctly 7594
- fix(scroll): scrolling animation and finger dragging conflict during continuous sliding, causing flickering 7522
- fix(pxp,vglite): extend base structure to create vglite and pxp draw units 7578

- fix(label): add assignment to uninitialized variable [7587](#)
- fix(xml): return if XMLs can't be saved [7588](#)
- fix(sysmon): fix heap memory overflow [7576](#)
- fix(scroll): handle scroll chain with snapping too [7491](#)
- fix(widgets): fix an incorrect text replacement [7548](#)
- fix(label): limit self size by max\_height style [7542](#)
- fix(widget\_chart): add assignment to uninitialized variable [7561](#)
- fix(label): fix updating scrolling label text [7533](#)
- fix(style): fix conditional jump or move depends on uninitialised value [7571](#)
- fix(wayland): delete the allocated display and buffers [7572](#)
- fix(docs): clarify how to pass color index... [7554](#)
- fix(lv\_canvas.c): fix use of uninitialized variable [7556](#)
- fix(draw\_buf): flush D-Cache after clear buffer [7550](#)
- fix(circle\_buff): replace a none ASCII character [7538](#)
- fix(lv\_freetype): clean up includes [7524](#)
- fix(tree): add NULL check [7526](#)
- fix(anim): remove the redefinition of lv\_anim\_set\_time [7500](#)
- fix(roller): Adjust the position of the lv\_roller\_set\_str function in lv\_roller. c and lv\_roller. h [7506](#)
- fix(font): fix get glyph id bug. [7404](#)
- fix(ap): fix ap unicode 0622+0644 [7482](#)
- fix(sysmon): fix crash when enable LV\_USE\_PERF\_MONITOR and nuttx backend [7483](#)
- fix(event): remove preprocess flag when get name [7468](#)
- fix(table): mark clicks as 'Invalid' when clicking on empty spaces [7153](#)
- fix(examples): fix infinite scroll crash [7459](#)
- fix(draw\_sw\_img): avoid divide by zero [7447](#)
- fix(vg\_lite): fix path memory reallocation error [7466](#)
- fix(dma2d): Add return value to DMA2d conversion functions [7456](#)
- fix(draw\_sw\_arc): add ASSERT\_MALLOC check [7448](#)
- fix(draw\_rect): refactor to insure header is initialized [7446](#)
- fix(draw\_sw\_triangle): add null check [7449](#)
- fix(vg\_lite): remove unnecessary grad image checks [7443](#)
- fix(vg\_lite): fix corner case of bar drawing [7441](#)
- fix(osal/pthread): add missing pthread\_attr\_destroy call [7434](#)
- fix(vg\_lite): fix stroke crash when update failed [7399](#)
- fix(docs): adjust colors to address accessibility issues [7409](#)
- fix(cmake): fix msvc building error [7401](#)
- fix(ebike-demo): make it compile with Arduino [7397](#)
- fix(test\_calendar.c): fix function name change from PR #7340 [7375](#)
- fix(draw\_sw\_fill): add NULL checks on grad [7355](#)

- fix(arduino): allow including lvgl\_private.h even if the examples and demos are in the src folder 7366
- fix(draw\_sw): wrong image buffer calculation 7387
- fix(scripts): fix image conversion was scrambling palette data 7367
- fix(scale): fix angle calculation error 7362
- fix(draw\_sw\_box\_shadow): add ASSERT\_MALLOC check 7344
- fix(docs): add missing quotes 7359
- fix(theme\_default): fix omitted style for selected text in `lv\_textarea` 7322
- fix(draw): fix minor maintenance issue in lv\_draw\_label.c 7296
- fix(imagebutton): warn if middle image is not set 7224
- fix(init): adjust the freetype initial order 7363
- fix(indev): prevent division by zero 7354
- fix(nxp-vglite): add missing '{' 7365
- fix(file\_explorer): quick access prototype outside guard 7356
- fix(lv\_conf): get rid of the LV\_CONF path building macros 7335
- fix(event): record rendering done moment after rendering finished 7228
- fix(draw\_sw): in lv\_draw\_sw\_rotate enable ARGB8888 functions for XRGB8888 too 7185
- fix(obj): use LV\_ASSERT\_NULL if LV\_ASSERT\_OBJ not enabled 7339
- fix(docs): stray space in style\_api\_gen.py and generated doc... 7309
- fix(issue): fix the platform input box not displaying fully 7316
- fix(style): missing (void\*) cast in macro LV\_STYLE\_CONST\_INIT 7304
- fix(flex): make min-width, grow, and wrap work together 7168
- fix(dave2d): required when using LVGL with LV\_COLOR\_DEPTH 32 7323
- fix(dave2d): fix implicit function definitions 7320
- fix(ffmpeg): add native filesystem API support for FFmpeg image decoder 7253
- fix(drivers/x11): fix unknown typename lv\_img\_dsc\_t 7294
- fix(ebike): add guards to the ebike demo translations 7319
- fix(vg\_lite): fix vector rendering missing image opa processing 7293
- fix(script): add ending for raw loader of ThorVG 7186
- fix(draw): fix incorrect clip area computation when clearing transparent framebuffers 7269
- fix(file\_explorer): fix path issues when returning to open other folders after opening a certain folder 7258
- fix(demo): resolve include when repository name is not 'lvgl' 7261
- fix(file\_explorer): table cell use after free 7239
- fix(roller): fix typo in scaling; prevent division by zero 7263
- fix(init): remove double call to lv\_draw\_sw\_deinit 7266
- fix(indev): don't reset all indevs if an object is set disabled 7216
- fix(obj): fix crash with LV\_SIZE\_CONTENT parent and % positioned child 7041
- fix(draw): fix bug introduced by PR #6638 7264
- fix(sw\_blend): add null pointer check for mask\_area 7251

- fix(lv\_draw\_sw\_line): fix lv\_draw\_line function causes a crash. 7248
- fix(buttonmatrix): add check for indev POINTER | BUTTON type 7125
- fix(freertos): compilation warning in non-IDF environments 7221
- fix(event) remove redundant last check 7227
- fix(cmake): fix installation with cmake --install 7161
- fix(chart): fix divide-by-zero and cursor error 7211
- fix(windows): Update document for Windows backend to clarify some features 7197
- fix(docs): fix bad highlight color 7199
- fix(chart): `lv\_malloc()` was allocating double size needed 7200
- fix(refr): avoid division by zero 7205
- fix(buttonmatrix): fix 1-off error in `lv\_buttonmatrix\_get\_button\_text()` 7193
- fix(scale): correct scale\_set\_arc\_properties function 7113
- fix(anim): fix user after free if the anim. is delete in the exec\_cb 7173
- fix(cmake): generate lvgl.pc in CMAKE\_CURRENT\_BINARY\_DIR 7127
- fix(demo): fix ebike demo header file inclusion issue 7133
- fix(table): fixed NULL pointer reference 7042
- fix(docs): trailing underscore looked like hyperlink to Sphinx 7172
- fix(docs): link in /lvgl/README.md to supported display types 7167
- fix(docs): fix broken links in ./lvgl/README.md 7146
- fix(file\_explorer): fix navigation handling 7124
- fix(neon): use conventional macro syntax 6887
- fix(docs): fix incorrect title and filename change for vg\_lite.rst 7148
- fix(draw): add assertion checks for malloc return values 7149
- fix(script): do not add pad if 'araw\_data\_len' can divide 'bblk\_size' evenly. 7109
- fix(tile) set minimum tiles to 1 if there is no SW render unit 7130
- fix(ebike): make it work without lottie too 7135
- fix(dma2d): add missing include for descriptor structs 7122
- fix(vg\_lite): fix linear gradient matrix error 7110
- fix(vg\_lite): fix arc drawing boundary case drawing error 7107
- fix(bin\_decoder): fix bug in handling premultiplied alpha flag 7106
- fix(tests): fix vg-lite buf address not aligned 7049
- fix(gdb): fix style prop getting 7088
- fix(obj\_scroll): include handle throwing animation in scrolling judgment 7011
- fix(tests): fix svg test assert and add missing ref images 7079
- fix(examples): fix font manager example build error in docs 7085
- fix(docs/README.md): reworked to handle several things 6992
- fix(codespace): use lv\_conf.defaults to generate lv\_conf.h 7075
- fix(vg\_lite): fix thorvg 32bit rendering test coordinate calculation overflow 7052
- fix(independent\_heap): add independent heap enable option 6953

- fix(anim\_timeline): run animations which do not have an exec cb 7043
- fix(vg\_lite): fix path bonding box coordinate overflow 7037
- fix(vg\_lite\_tvg): fix shape\_set\_stroke calling order error 7039
- fix(dave2d): fix dave2d private include issue 7016
- fix(anim): compensate over time 6989
- fix(docs): fix error blocking API-doc generation under Windows 6990
- fix(scale): horizontal scale sections not visible 6982
- fix(vg\_lite\_tvg): fix int32 type mismatch 7022
- fix(png): move png\_image\_free to just before return NULL 7020
- fix(png): fix mem leak in libpng decoder 6952
- fix(drivers): add missing includes 6905
- fix(benchmark): improve responsiveness and use the LVGL logo instead of the cogwheel 6980
- fix(drivers): calculate stride align correctly 6976
- fix(rt-thread): fix compile error 6938
- fix(grid): fix naming conflicts 6963
- fix(thorvg): fix clipped clippers 6956
- fix(docs): restore missing on-line examples 6927
- fix(docs): fix most sphinx warnings 6916
- fix(layout): calculate content width using x alignment 6948
- fix(style): remove transitions when a local style property is set 6941
- fix(docs): eliminate 2 types of sphinx warnings: 6928
- fix: eliminate misc sphinx warnings... 6929
- fix(calendar): fix lv\_calendar\_gregorian\_to\_chinese compile error 6894
- fix(fs): add lv\_fs\_dir\_t to lv\_fs.h 6925
- fix(indev): fix hovering disabled obj resets indev 6855
- fix(gif): added bounds check in gif decoder 6863
- fix(freertos): sync signal from isr fixed 6793
- fix(freertos): use xSemaphoreTakeRecursive 6803
- fix.dropdown): automatically center dropdown content 6881
- fix(draw): fix sw compile error when disable LV\_DRAW\_SW\_COMPLEX 6895
- fix(textarea): fix placeholder text cannot be centered (#6879) 6917
- fix(libinput): private headers 6869
- fix(color): add missing ARGB8565 alpha check 6891
- fix(Kconfig): Fix non existent LV\_STDLIB\_BUILTIN 6851
- fix(display): remove the unused sw\_rotate field 6866
- fix(image): lv\_image\_set\_inner\_align() behaviour with LV\_IMAGE\_ALIGN\_STRETCH 6864
- fix(bar): fix bit overflow 6841
- fix(indev): don't wait until release when a new object is found 6831
- fix(cmake): fix install 6787

- fix(docbuild): update style doc to reflect Doxygen needs 6705
- fix(vg\_lite): fix image transform clipping area error 6810
- fix(glfw/opengles): fix buf\_size calculation error 6830
- fix(roller): fix stringop overflow 6826
- fix(perf): perf monitor FPS 6798
- fix(micropython): missing bidi private header feature guard 6801
- fix(draw): fix artifact when rotating ARGB8888 images 6794
- fix(sdl): check against NULL before using the driver data of a display 6799
- fix(assets): add missing strides 6790
- fix(arc): ignore hits that are outside drawn background arc 6753
- fix(vg\_lite): fixed clip\_radius image cropping error 6780
- fix(vg\_lite/vector): convert gradient matrix to global matrix 6577
- fix(spangroup): fix height calculation error 6775
- fix(buttonmatrix): use const arrays 6765
- fix(ime): fix ime crash when input\_char is too long 6767
- fix(script): follow lv\_conf\_template.h changes in generate\_lv\_conf.py 6769
- fix(vg\_lite): select blend mode based on premultiplication 6766
- fix(docbuild): reformat comments for Doxygen in `lv\_conf\_template.h` 6673
- fix(draw): cast color\_format in LV\_DRAW\_BUF\_INIT\_STATIC 6729
- fix(sdl): nested comment is not allowed 6748
- fix(ime\_pinyin): fix letter count wrong when using some dictionary 6752
- fix(anim): use correct variable `repeat\_cnt` 6757
- fix(xml): add missing style properties d2ed41d
- fix(font): impore static bitmap handling with stride ccbbfcc
- fix(vglite) upstream comments a2b687c
- fix(label): use LV\_LABEL\_LONG\_MODE\_\* instead of LV\_LABEL\_LONG\_\* db11e7b
- fix(vglite) do not apply scissor cut if border is complete 29e3da2
- fix(refr): if tile\_cnt == 1 don't create new layers. d5b02fe
- fix(sdl): fix caching with absolute coordinates d469d65
- fix(xml): use opa type for opacity reapted style props 21e3350
- fix(nxp) coverity issues a107090
- fix(xml): handle nested 'extends' handling with components 1964dfe
- fix(xml): minor fixes of typos and cleanups c0a7eff
- fix(vglite) add VGLITE\_CHECK\_ERROR for all vg\_lite functions 69f7599
- fix(label): improve the position keeping of updated scrolling texts aff9ad3
- fix(xml): fix xml property propagation issues 5652914
- fix(chart): fix saving the address of a local variable in lv\_example\_chart\_3() 37ea110
- fix(xml\_style): add min/max\_width/height and recolor 4518d34
- fix(event): allow using components without registering events 22e0ebb

- fix(vglite) Do not dispatch any task if draw unit is busy with "wait\_for\_finish". b89958a
- fix(vglite) update triangle and line dsc to align with upstream changes 0a0fad4
- fix(vglite) add checks for alignment, stride and static bitmap 5564b5b
- fix(xml): handle not existing subjects gracefully 0b44606
- fix(NemaGFX): arc angles 6492096
- fix(xml): fix typos in lv\_bar.xml and lv\_slider.xml 59bc690
- fix(draw): do not dispatch if no unit took the task 3c07bd9
- fix(font): add the missing flush cache operation dcf934d
- fix(xml): make the XML parser work on MCUs too 2e7d17f
- fix(nuttx\_fbdev): fix memory buffer handling b662802
- fix(xml): fix enum type for lv\_obj bind\_state\_if 859b2e8
- fix(chart): add more error handling in XML 7256ba0
- fix(xml): add missing align and layout values fc4f8fa
- fix(nxp) solve all build warnings be9b6a2
- fix(xml): support <view ...> 1d71728
- fix(ffmpeg): clear the first image data eecebd2
- fix(anim): make sure the resumed animation doesn't skip the first update eeb1fef
- fix(vglite) ARGB8565 color format support 284e7a4
- fix(benchmark): better handle font dependencies 0bbded7
- fix(nuttx/cache): add invalidate\_cache and flush\_cache into image\_cache\_draw\_buf\_handlers ea538de
- fix(pxp) disable pxp draw tile image d6850a8
- fix(vglite) initialize the paths empty 890cec7
- fix(observer): fix 2 warnings emitted by VS C compiler... bae25ad
- fix(vglite) run without DRAW\_THREAD 5426204
- fix(xml): fix roller's XML to reflect better on the C API b27fe9d
- fix(observer.c): fix inconsistency of whitespace around conditional directives c28c7e8
- fix(scale): fix typo in XML ea18026
- fix(xml): chart and slider minor fixes ed7c3be
- fix(xml): fix widget xmls ecfe334
- fix(vglite) fix stride calculation 9e5039c
- fix(vglite) skip VG\_LITE\_PATTERN\_REPEAT tasks on GC255 655f744
- fix(pxp) draw\_img src\_area parameter 33de407
- fix(demo): Add missing Kconfig symbols for ebike and high res 7d088e2
- fix(sdl): make it work with canvas rendering too 5d9064c
- fix(draw): fix not reallocating draw\_buf-s for glyphs 2d0f6be
- fix(vglite) do not draw tiled images when using blit split 9e9d821
- fix(vglite) draw\_pattern starts the GPU after setting the command buffer a23bdbd
- fix(sw\_blend) make sure mask\_stride is initialized 86d4385
- fix(core):fix overflow\_const issue 1dc4096

- fix(vglite) update get\_glyph function for static bitmaps [18cdb6a](#)
- fix(obj\_tree): support NULL parent in lv\_obj\_find\_by\_name [2e5931b](#)
- fix(test): print n-s in logs correctly [f317067](#)
- fix(span): fix the type of the span element [9b061e8](#)
- fix(vglite) vglite draw thread stack size increase for gcFEATURE\_VG\_SIMPLIFIED\_BEZIER [b67bfee](#)
- fix(ebike): fix include path [255352b](#)
- fix(draw): init mask\_stride when sw draw triangle [15db6f0](#)
- fix(table): refresh cell size after ctrl change [47f3c65](#)
- fix(obj\_style): fix transition anim of style\_recolor [f63b275](#)
- fix(style): skip figma\_node\_id attribute [4b23bae](#)
- fix(ffmpeg): free packet [3605c32](#)
- fix(observer): protect against possible negative arg value [43b3234](#)
- fix(font): allow non-constant LV\_FONT\_DEFAULT again (fixes #7788) (#8059) [9f090d2](#)
- feat(obj): add obj name support [626d6b5](#)
- fix(observer): fix bug in `lv\_obj\_bind\_checked()` function [02c3d7c](#)
- fix(event): consider NULL as a wildcard for callback arg in lv\_obj\_remove\_event\_cb\_with\_user\_data() [0ae888b](#)
- fix: add +1 but for enum bitfields to avoid overflow to negative [253981a](#)
- fix(sdl): add static\_text flag to to lv\_draw\_label\_dsc\_t [7808070](#)

### 13.1.6 Esempi

- example(keyboard): add draw event example to customize each button [8172](#)
- example(freetype): fix docs build error [8001](#)
- example(checkbox): Correct checkbox enable comment [7763](#)

### 13.1.7 Docs

- docs(coordinates.rst): fix erroneous function... [8283](#)
- docs(Renesas): add docs for Renesas RZ/A3M [8285](#)
- docs(XML): proofread XML docs (batch 2 of 2) [8185](#)
- docs(NuttX): Add more NuttX docs [8235](#)
- docs(ft81x.rst): move under ./docs/src/ directory [8231](#)
- docs(Renesas): make "Supported Boards" table cells narrower [8214](#)
- docs(misc fixes): fix misc items (details below)... [8186](#)
- docs(Renesas): Renesas G2UL docs change video link [8183](#)
- docs(XML): proofread XML docs (batch 1 of 2) [8184](#)
- docs: fix Doxygen parsing of attribute prefixes [8179](#)
- docs(doxxygen): exclude deprecated CJK fonts for Doxygen. [8141](#)
- docs(table css): fix table formatting... [8196](#)
- docs(stm32): Fix errors in FreeRTOS Example Code [8163](#)
- docs(conf.py): fix URL path to viewing doc source on GitHub [8167](#)

- **docs(Renesas): Reorganize Renesas docs** [8153](#)
- **docs(3dtexture.rst): get this doc into TOC and proofread** [8143](#)
- **docs(viewe): Update docs** [8103](#)
- **docs(gdb\_plugin): add missing info draw\_unit command** [8078](#)
- **docs: add missing pcpp dependency** [8069](#)
- **docs(introduction.rst): fix bug in API links...** [8049](#)
- **docs(observer.rst): remove now-unnecessary warnings** [8048](#)
- **docs(build.py): enhance build-output readability** [8028](#)
- **docs(doc-gen): eliminate several breathe errors** [8036](#)
- **docs(misc): misc clean-up...** [8016](#)
- **docs(draw\_layers): combine draw-layers topics...** [8034](#)
- **docs(doc-gen): eliminate all non-API-page errors** [8035](#)
- **docs(sdl): Add SDL docs** [8023](#)
- **docs(draw.rst): restructure and proofread from PR #7241** [7976](#)
- **docs(timer): document LV\_NO\_TIMER\_READY** [7920](#)
- **docs(README\_zh.md): add missing examples section** [7897](#)
- **docs: remove non-ASCII bytes -- batch 1** [7974](#)
- **docs: fix erroneous api links** [7967](#)
- **docs(getting\_started.rst): fix typos (replaces PR #7829)** [7970](#)
- **docs: fix previously-broken api links** [7971](#)
- **docs(readme): fix broken links** [7961](#)
- **docs(doc\_builder.py): overhaul, remove bugs, clean up, modularize** [7913](#)
- **docs(test.rst): proofread test doc, minor edits** [7906](#)
- **docs(viewe): add Viewe docs** [7916](#)
- **docs(test): fix the location of test.rst** [7900](#)
- **docs: fix doc-build failure** [7879](#)
- **docs(restructure): restructure TOC and contents for ease of use** [7847](#)
- **docs(introduction): update active state of v8.3** [7867](#)
- **docs(boards): proofread batch 21** [7769](#)
- **docs: convert 4 `rst` files back to `md` files since not part of doc-gen** [7839](#)
- **docs(misc): pre-re-org wrap-up...** [7804](#)
- **docs(bindings): proofread batch 20** [7764](#)
- **docs(xml): fix Sphinx errors in XML-component docs** [7820](#)
- **docs(libs): proofread docs batch 19** [7709](#)
- **docs(intro): fix typo** [7812](#)
- **docs(libs): proofread docs batch18** [7688](#)
- **docs(roller): fix roller API-docs spelling error** [7735](#)
- **docs(xml): fix not using the docs** [7785](#)
- **docs(event): add warning not to modify Widgets during draw events.** [7685](#)

- **docs(licenses): add the missing license files** 7668
- **docs(fbdev): add information about resolution issue** 7025
- **docs(Alif): add docs for Alif chip vendor** 7622
- **docs: general typo fix on the NXP related docs.** 7626
- **docs(Riverdi): Add Riverdi board docs** 7629
- **docs(fs): pin diagram dependency version** 7453
- **docs: clarify what C functions are used in storage operations** 7627
- **docs(timer): fix usage of timer user data** 7575
- **docs(arduino): add Arduino Giga Display Shield reference** 7584
- **docs(layer): fix layer rst function error** 7541
- **docs(label): add documentation and example for text recolor** 7460
- **docs(intro): fix typos** 7485
- **docs(details): fix typos** 7484
- **docs(profiler): remove tick unit** 7433
- **docs(disp): add rotation examples** 7314
- **docs: chinese translation link only on the homepage** 7370
- **docs(buildroot): add guide to build and embed lvgl on br image** 7084
- **docs(font): add a typesetting section** 7351
- **docs: update LV\_OBJ\_FLAG\_OVERFLOW\_VISIBLE usage** 7101
- **docs: update api\_json.rst** 7318
- **docs(display): correct spelling** 7225
- **docs: fix typo in Introduction: "similator" to "simulator"** 7162
- **docs(driver): fix link to example code in gen\_mipi.rst** 7128
- **docs(font\_manager): add documentation and test cases** 7058
- **docs(scale): fix rst** 7045
- **docs(license): add font license files** 6971
- **docs(yocto): add guide to integrate lvgl recipe in Yocto** 7024
- **docs: fix broken links** 6910
- **docs: update macro LV\_TA\_CURSOR\_LAST => LV\_TEXTAREA\_CURSOR\_LAST** 6995
- **docs(display): mention how to manipulate the invalidated area** 6836
- **docs(event): minor naming fixes** 6939
- **docs: fix typo in introduction** 6871
- **docs(os): fix formatting** 6829
- **docs(arm): add to index and update Arm2D docs** 6768
- **docs(xml): reorganize and extend the XML docs** ad8a9d9
- **docs(observer): proofread/edit/clarify `observer.rst`** efcdff7
- **docs(observer): re-work API (and some internal) documentation** a0e37a3
- **docs(index): sort the index links alphabetically** 032c2c3
- **docs(iter): add docstrings for lv\_iter\_t** 189102d

- **docs(xml): document events** [4dfdd18](#)
- **docs(observer.rst): break up so internal TOC is...** [0944979](#)
- **docs(API): make API TOCs easier to use** [b5dc041](#)
- **docs(ffmpeg): images always loaded with lvgl fs** [850426b](#)
- **docs(index): complete the missing link in the index** [4d929f3](#)

### 13.1.8 CI and tests

- **test(screenshot): use lodepng instead of libpng** [8275](#)
- **ci: build examples with c++ compiler** [8261](#)
- **ci: move docs build config to main repo** [8137](#)
- **ci(release\_updater): Add bi-weekly schedule. Only update master branches on schedule** [8074](#)
- **ci(docs): remove concurrency clause** [8102](#)
- **ci(docs): fix commit ref when workflow is triggered by master push** [8100](#)
- **ci: build docs on pr** [8089](#)
- **ci(micropython): fix MicroPython github build action** [7945](#)
- **ci(Windows): vcpkg url patch** [7948](#)
- **ci(ubuntu): ci use ubuntu24.04** [6241](#)
- **ci: add 8bit build option** [7574](#)
- **ci: fix port release branch updater git config command** [7722](#)
- **ci: Port release updater update master as well** [7650](#)
- **ci: cancel previous workflow run on new commit** [7619](#)
- **ci: add port release updater** [7590](#)
- **ci(esp): add esp32s3 build workflow** [7242](#)
- **ci(ref\_imgs): make the CI fail if there not all ref. images are added** [7086](#)
- **ci(Kconfig): error on leading spaces (should be tabs)** [92d424e](#)
- **ci(ccpp): add --auto-clean flag to lower disk space usage** [970ca51](#)
- **ci: fix a reference image** [1506779](#)

### 13.1.9 Others

- **chore(cmsis-pack): update cmsis-pack for v9.3.0 release** [8113](#)
- **revert.dropdown): add lv\_anim\_enable\_t parameter to lv\_dropdown\_set\_selected (#7310)** [8304](#)
- **chore: use keyword arguments in regex calls to avoid warnings** [8288](#)
- **chore: remove extraneous file from repository** [8284](#)
- **build(makefile): add class, instance and xml** [8233](#)
- **build: restructure cmake** [8210](#)
- **chore(draw/sw): minor cleanup** [8083](#)
- **refactor(cache): rename and restructure cache-related files** [8148](#)
- **chore(profiler): add built-in profiler default enable config** [8120](#)
- **doc(freetype) add ftstroke.c to example Makefile** [8114](#)
- **Update basics.rst to fix wrong variable name** [8112](#)

- **arch(font\_manager): add multiple font backend support** 8038
- **build: remove pcpp dependency** 8090
- **chore(uefi): fix typos** 8094
- **arch(demos): move new demos to lv\_demos** 8022
- **build(custom): fix include path of lv\_conf.cmake** 8052
- **arch(cmake): add native Kconfig support to cmake** 7934
- **Bugfix and improvement for the logging** 8025
- **chore(.gitignore): move tests directory ignores to ./tests/.gitignore** 7995
- **chore(bin\_decoder): add check for image source type in `decode\_compressed`** 7980
- **refact(vg\_lite): refactoring vector rendering** 7928
- **chore(Kconfig): set vg\_lite stroke default cache size to 8** 7939
- **chore(deps): bump JamesIves/github-pages-deploy-action from 4.7.2 to 4.7.3** 7873
- **chore(deps): bump espressif/upload-components-ci-action from 1 to 2** 7874
- **Dont's squash - gradient updates** 7646
- **chore(lv\_conf\_template): replace tab with space** 7736
- **chore(obj): make lv\_obj\_get\_scroll\_{left,right,top,bottom}() const** 7674
- **Feat(docs): proofread docs batch17** 7680
- **refactor(draw): use task instead of draw\_unit as argument to draw functions** 7240
- **chore(draw): fix unused function warning** 7611
- **chore(deps): bump JamesIves/github-pages-deploy-action from 4.7.1 to 4.7.2** 7534
- **chore(deps): bump carlosperate/arm-none-eabi-gcc-action from 1.9.1 to 1.10.0** 7535
- **chore: fix typos in libs** 7516
- **chore(pr-template): add 'Fixes' to the template** 7462
- **chore: fix typos in lv\_draw\_label.h** 7511
- **chore: fix typos in lv\_display.h** 7512
- **chore: fix typos in lv\_text.c** 7513
- **chore: fix typos in others** 7514
- **chore: fix typos in drivers** 7517
- **chore: fix typos in some header files** 7518
- **chore: fix typos in nema\_gfx** 7519
- **chore: fix typos in core** 7510
- **chore(tests): assert array size with LV\_ARRAY\_DEFAULT\_CAPACITY** 7497
- **chore(demos): fix spelling error in demos** 7479
- **chore(indev\_gesture): add missing config and fix warning** 7461
- **chore(deps): bump JamesIves/github-pages-deploy-action from 4.6.8 to 4.7.1** 7392
- **chore(display.rst): fix typo** 7376
- **chore(example\_tabview): fix typo** 7377
- **chore: removes BOM** 6961

- **chore(draw\_sw): add warning if the required color format is not enabled during image transformations** 7315
- **chore(nema): fix build warning** 7301
- **refactor(draw): extract function to improve readability** 7229
- **chore(api): prevent API leaks (cache and thread)** 7220
- **Feature/nema updates** 7245
- **chore(NemaGFX): add missing header, correct the docs, provided HAL is optional** 7174
- **chore(deps): bump arduino/arduino-lint-action from 1 to 2** 7209
- **chore(deps): bump actions/upload-artifact from 3 to 4** 7210
- **chore(cmsis-pack): ensure the v9.2.2 cmsis-pack is visible on keil.arm.com** 7201
- **chore(fs): treat '/' as valid driver letter** 6986
- **chore(obj\_scroll): improve function documentation** 7150
- **chore(svg): LV\_USE\_SVG: document and enforce dependency on LV\_USE\_VECTOR\_GRAPHIC** 7120
- **refactor(gdb): refactor gdb Debugger to be a gdb command** 7140
- **chore(vg\_lite): fix path bounding\_box typo** 7134
- **refactor(gdb): refactor gdb script** 7123
- **chore(demos/ebike): fixed chinese translation errors** 7102
- **chore(cmsis-pack): update cmsis pack for v9.3.0-dev** 7080
- **chore(qrcode): replaced libc includes** 7077
- **chore(docs): fix font manager example index typo** 7083
- **chore: fix minor compile warnings** 6987
- **chore(test): add missing label\_recolor ref images** 7064
- **chore: remove UTF-8 BOMs** 6942
- **chore(stale): use the correct labels** 7017
- **chore(deps): bump JamesIves/github-pages-deploy-action from 4.6.3 to 4.6.8** 6984
- **chore: use the new labels in GitHub actions and docs** 6940
- **chore(lv\_conf\_internal): avoid LV\_CONF\_SKIP redefinition warning** 6960
- **chore: fix compile warnings** 6817
- **chore(widgets): add comments to the tileview and win** 6856
- **chore(vg\_lite): remove unnecessary buffer checks** 6921
- **chore(Kconfig): add version info to Kconfig file to check mismatch** 6789
- **chore(array): move array typedef to lv\_types.h** 6850
- **chore: add underscore prefix for names after struct and enum** 6688
- **chore(scripts): allow to run update\_version.py with python3.8** 6788
- **chore(font\_manager): update to v9.1 API** 6776
- **chore(deps): bump carlosperate/arm-none-eabi-gcc-action from 1.9.0 to 1.9.1** 6779
- **chore(tests): fix build break on macOS** 6773
- **chore: bump version to v9.3.0-dev** 6764
- **Revert "perf(draw): skip empty draw tasks (#6720)"** 6758

- **chore(NemaGFX): update library** [7232ad7](#)
- **refactor(xml): rename ctx to scope as it better describes its meaning** [8e19bb4](#)
- **refactor(vglite) use dynamic paths for drawing** [e1ccefa](#)
- **refactor(chart): add 'series' keyword in value setters/getters for consistency** [5f2888d](#)
- **refactor(obj): use lv\_obj\_set\_flag instead of lv\_obj\_update\_flag** [3b394ac](#)
- **refactor(vglite) use dynamic path\_data for drawing** [4fab9de](#)
- **refactor(vglite) remove vg from variable names** [90fa702](#)
- **chore(class): use lv\_ prefixed s to avoid naming conflicts with the user widgets** [6d99933](#)
- **refactor(vglite) remove unnecessary arguments of draw functions** [696693b](#)
- **refactor(iter): use circle\_buf to simplify codes** [b3bf2ab](#)
- **refactor(objid): rename lv\_obj\_get\_child\_by\_id to lv\_obj\_find\_by\_id for consistency** [56c44bf](#)
- **chore(iter): add more safe checker and const qualifier** [23add00](#)
- **refactor(vglite) use dynamic gradients for drawing** [fc43e07](#)
- **chore(doc): add new debugging folder** [a0d3efe](#)
- **refactor(table): convert lv\_table\_add\_cell\_ctrl to a setter** [39c4011](#)
- **refactor(pxp) remove unnecessary arguments of draw functions** [7a7edb5](#)
- **revert(xml): revert the accidentally modified view.xml** [09d0d10](#)
- **refactor(vglite) fix NON ASYNC mode** [8b56df4](#)
- **chore(xml): minor fixes** [3523270](#)
- **chore(anim): define anim on/off as true/false** [2d8ff35](#)
- **chore(obj): deprecate lv\_obj\_find\_by\_id in favor of lv\_obj\_find\_by\_name** [346395d](#)
- **chore(xml): fix warning** [868d7b5](#)
- **chore(obj): add style\_border\_side XML support** [c65eaf2](#)
- **refactor(chart): add 'series' keyword in value setters/getters for consistency** [e90cfb7](#)
- **chore(tests): Enable -Wunused-function** [b6adc8f](#)

### 14.1 lv\_api\_map\_v8.h

#### Defines

```
LV_DISP_ROTATION_0  
LV_DISP_ROTATION_90  
LV_DISP_ROTATION_180  
LV_DISP_ROTATION_270  
LV_DISP_RENDER_MODE_PARTIAL  
LV_DISP_RENDER_MODE_DIRECT  
LV_DISP_RENDER_MODE_FULL  
LV_BTNMATRIX_BTN_NONE  
LV_BTNMATRIX_CTRL_HIDDEN  
LV_BTNMATRIX_CTRL_NO_REPEAT  
LV_BTNMATRIX_CTRL_DISABLED  
LV_BTNMATRIX_CTRL_CHECKABLE
```

**LV\_BTNMATRIX\_CTRL\_CHECKED**

**LV\_BTNMATRIX\_CTRL\_CLICK\_TRIG**

**LV\_BTNMATRIX\_CTRL\_POPOVER**

**LV\_BTNMATRIX\_CTRL\_CUSTOM\_1**

**LV\_BTNMATRIX\_CTRL\_CUSTOM\_2**

**LV\_IMGBTN\_STATE\_RELEASED**

**LV\_IMGBTN\_STATE\_PRESSED**

**LV\_IMGBTN\_STATE\_DISABLED**

**LV\_IMGBTN\_STATE\_CHECKED\_RELEASED**

**LV\_IMGBTN\_STATE\_CHECKED\_PRESSED**

**LV\_IMGBTN\_STATE\_CHECKED\_DISABLED**

**LV\_RES\_OK**

**LV\_RES\_INV**

**LV\_INDEV\_STATE\_PR**

**LV\_INDEV\_STATE\_REL**

**lv\_obj\_del**

**lv\_obj\_del\_async**

**lv\_obj\_clear\_flag**

**lv\_obj\_clear\_state**

**lv\_indev\_set\_disp**

**lv\_indev\_get\_act**

**lv\_scr\_act**

**lv\_disp\_remove**

**lv\_disp\_set\_default**  
**lv\_disp\_get\_default**  
**lv\_disp\_get\_next**  
**lv\_disp\_set\_rotation**  
**lv\_disp\_get\_hor\_res**  
**lv\_disp\_get\_ver\_res**  
**lv\_disp\_get\_physical\_hor\_res**  
**lv\_disp\_get\_physical\_ver\_res**  
**lv\_disp\_get\_offset\_x**  
**lv\_disp\_get\_offset\_y**  
**lv\_disp\_get\_rotation**  
**lv\_disp\_get\_dpi**  
**lv\_disp\_get\_antialiasing**  
**lv\_disp\_flush\_ready**  
**lv\_disp\_flush\_is\_last**  
**lv\_disp\_get\_scr\_act**  
**lv\_disp\_get\_scr\_prev**  
**lv\_disp\_load\_scr**  
**lv\_scr\_load**  
**lv\_scr\_load\_anim**  
**lv\_disp\_get\_layer\_top**  
**lv\_disp\_get\_layer\_sys**  
**lv\_disp\_send\_event**

`lv_disp_set_theme`  
`lv_disp_get_theme`  
`lv_disp_get_inactive_time`  
`lv_disp_trig_activity`  
`lv_disp_enable_invalidation`  
`lv_disp_is_invalidation_enabled`  
`lv_disp_refr_timer`  
`lv_disp_get_refr_timer`  
`lv_timer_del`  
`lv_anim_del`  
`lv_anim_del_all`  
`lv_anim_set_ready_cb`  
`lv_group_del`  
`lv_txt_get_size`  
`lv_txt_get_width`  
`lv_img_create`  
`lv_img_set_src`  
`lv_img_set_offset_x`  
`lv_img_set_offset_y`  
`lv_img_set_angle`  
`lv_img_set_pivot`  
`lv_img_set_zoom`  
`lv_img_set_antialias`

```
lv_img_get_src  
lv_img_get_offset_x  
lv_img_get_offset_y  
lv_img_get_angle  
lv_img_get_pivot  
lv_img_get_zoom  
lv_img_get_antialias  
lv_imgbtn_create  
lv_imgbtn_set_src  
lv_imgbtn_set_state  
lv_imgbtn_get_src_left  
lv_imgbtn_get_src_middle  
lv_imgbtn_get_src_right  
lv_list_set_btn_text  
lv_list_get_btn_text  
lv_list_add_btn  
lv_btn_create  
lv_btndmatrix_create  
lv_btndmatrix_set_map  
lv_btndmatrix_set_ctrl_map  
lv_btndmatrix_set_selected_btn  
lv_btndmatrix_set_btn_ctrl  
lv_btndmatrix_clear_btn_ctrl
```

```
lv_btnmatrix_set_btn_ctrl_all  
lv_btnmatrix_clear_btn_ctrl_all  
lv_btnmatrix_set_btn_width  
lv_btnmatrix_set_one_checked  
lv_btnmatrix_get_map  
lv_btnmatrix_get_selected_btn  
lv_btnmatrix_get_btn_text  
lv_btnmatrix_has_button_ctrl  
lv_btnmatrix_get_one_checked  
lv_tabview_get_tab_btns  
lv_tabview_get_tab_act  
lv_tabview_set_act  
lv_tileview_get_tile_act  
lv_obj_set_tile_id  
lv_obj_set_tile  
lv_roller_set_visible_row_cnt  
lv_roller_get_option_cnt  
lv_table_set_col_cnt  
lv_table_set_row_cnt  
lv_table_get_col_cnt  
lv_table_get_row_cnt  
lv_table_set_col_width  
lv_table_get_col_width
```

`lv_dropdown_get_option_cnt`

`lv_obj_get_child_cnt`

`lv_obj_get_disp`

`lv_obj_delete_anim_ready_cb`

`LV_STYLE_ANIM_TIME`

`LV_STYLE_IMG_OPA`

`LV_STYLE_IMG_RECOLOR`

`LV_STYLE_IMG_RECOLOR_OPA`

`LV_STYLE_SHADOW_OFS_X`

`LV_STYLE_SHADOW_OFS_Y`

`LV_STYLE_TRANSFORM_ANGLE`

`lv_obj_get_style_anim_time`

`lv_obj_get_style_img_opa`

`lv_obj_get_style_img_recolor`

`lv_obj_get_style_img_recolor_filtered`

`lv_obj_get_style_img_recolor_opa`

`lv_obj_get_style_shadow_ofs_x`

`lv_obj_get_style_shadow_ofs_y`

`lv_obj_get_style_transform_angle`

`lv_obj_get_style_bg_img_src`

`lv_obj_get_style_bg_img_recolor`

`lv_obj_get_style_bg_img_recolor_opa`

`lv_obj_set_style_anim_time`

```
lv_obj_set_style_img_opa  
  
lv_obj_set_style_img_recolor  
  
lv_obj_set_style_img_recolor_opa  
  
lv_obj_set_style_shadow_ofs_x  
  
lv_obj_set_style_shadow_ofs_y  
  
lv_obj_set_style_transform_zoom  
  
lv_obj_set_style_transform_angle  
  
lv_obj_set_style_bg_img_src  
  
lv_obj_set_style_bg_img_recolor  
  
lv_obj_set_style_bg_img_recolor_opa  
  
lv_style_set_anim_time  
  
lv_style_set_img_opa  
  
lv_style_set_img_recolor  
  
lv_style_set_img_recolor_opa  
  
lv_style_set_shadow_ofs_x  
  
lv_style_set_shadow_ofs_y  
  
lv_style_set_transform_angle  
  
lv_style_set_transform_zoom  
  
lv_style_set_bg_img_src  
  
lv_style_set_bg_img_recolor  
  
lv_style_set_bg_img_recolor_opa  
  
lv_keyboard_get_selected_btn  
  
lv_keyboard_get_btn_text
```

**LV\_ZOOM\_NONE**

**lv\_image\_decoder\_built\_in\_open**

**lv\_image\_decoder\_built\_in\_close**

**LV\_IMG\_DECLARE(var\_name)**

Use this macro to declare an image in a C file

## Typedef

typedef int32\_t **lv\_coord\_t**

typedef lv\_result\_t **lv\_res\_t**

typedef *lv\_image\_dsc\_t* **lv\_img\_dsc\_t**

typedef lv\_display\_t **lv\_disp\_t**

typedef *lv\_display\_rotation\_t* **lv\_disp\_rotation\_t**

typedef *lv\_display\_render\_mode\_t* **lv\_disp\_render\_t**

typedef *lv\_anim\_completed\_cb\_t* **lv\_anim\_ready\_cb\_t**

typedef *lv\_screen\_load\_anim\_t* **lv\_scr\_load\_anim\_t**

typedef *lv\_buttonmatrix\_ctrl\_t* **lv\_btnmatrix\_ctrl\_t**

## Functions

static inline uint32\_t **lv\_task\_handler**(void)

static inline void **lv\_obj\_move\_foreground**(lv\_obj\_t \*obj)

Move the object to the foreground. It will look like if it was created as the last child of its parent. It also means it can cover any of the siblings.

### Parametri

**obj** -- pointer to an object

static inline void **lv\_obj\_move\_background**(lv\_obj\_t \*obj)

Move the object to the background. It will look like if it was created as the first child of its parent. It also means any of the siblings can cover the object.

### Parametri

**obj** -- pointer to an object

## 14.2 lv\_api\_map\_v9\_0.h

### Defines

**lv\_image\_set\_align**

```
lv_image_get_align  
lv_button_bind_checked  
LV_DRAW_BUF_DEFINE  
  
_lv_utils_bsearch  
  
lv_draw_buf_align_user  
lv_draw_buf_create_user  
  
lv_draw_buf_width_to_stride_user  
  
lv_draw_buf_dup_user  
lv_draw_buf_invalidate_cache_user(handlers, drawbuf, area)  
lv_draw_buf_flush_cache_user(handlers, drawbuf, area)  
lv_draw_buf_destroy_user(handlers, drawbuf)
```

## 14.3 lv\_api\_map\_v9\_1.h

### Defines

```
_LV_EVENT_LAST  
  
_lv_event_code_t  
  
_lv_event_mark_deleted  
  
lv_obj_add_event  
  
_LV_STYLE_LAST_BUILT_IN_PROP  
  
_LV_FLEX_REVERSE  
  
_LV_FLEX_WRAP  
  
_LV_FLEX_COLUMN  
  
_lv_area_is_equal  
  
_lv_area_is_in  
  
_lv_area_intersect
```

\_lv\_area\_is\_point\_on  
  
\_lv\_area\_join  
  
\_lv\_image\_buf\_get\_transformed\_area  
  
\_lv\_ll\_init  
  
\_lv\_ll\_ins\_head  
  
\_lv\_ll\_ins\_prev  
  
\_lv\_ll\_ins\_tail  
  
\_lv\_ll\_get\_head  
  
\_lv\_ll\_get\_tail  
  
\_lv\_ll\_get\_next  
  
\_lv\_ll\_get\_prev  
  
\_lv\_ll\_get\_len  
  
\_lv\_ll\_move\_before  
  
\_lv\_ll\_is\_empty  
  
\_lv\_ll\_clear  
  
\_lv\_ll\_remove  
  
\_lv\_ll\_chg\_list  
  
\_LV\_LL\_READ  
  
\_LV\_LL\_READ\_BACK  
  
\_lv\_obj\_scroll\_by\_raw  
  
\_lv\_obj\_get\_ext\_draw\_size  
  
\_lv\_indev\_scroll\_handler  
  
\_lv\_display\_refr\_timer

`_lv_disp_refr_timer`  
`_lv_disp_get_refr_timer`  
`lv_obj_get_child_by_id`  
`lv_obj_update_flag`  
`_lv_inv_area`  
`lv_chart_set_all_value`  
`lv_calendar_set_showed_date`  
`lv_chart_set_range`  
`lv_chart_set_value_by_id`  
`lv_chart_get_x_array`  
`lv_chart_get_y_array`  
`lv_chart_set_ext_x_array`  
`lv_chart_set_ext_y_array`  
`LV_LABEL_LONG_WRAP`  
`LV_LABEL_LONG_DOT`  
`LV_LABEL_LONG_SCROLL`  
`LV_LABEL_LONG_SCROLL_CIRCULAR`  
`LV_LABEL_LONG_CLIP`  
`lv_anim_set_time`  
`lv_anim_set_playback_time`  
`lv_anim_set_playback_delay`  
`lv_anim_set_playback_duration`  
`lv_gradient_init_stops`

```
lv_gradient_stop_t  
lv_spangroup_new_span  
lv_spangroup_refr_mode  
lv_slider_set_left_value  
lv_calendar_header_arrow_create  
lv_calendar_header_dropdown_create
```

## 14.4 lv\_api\_map\_v9\_2.h

### Defines

```
LV_SCR_LOAD_ANIM_NONE  
LV_SCR_LOAD_ANIM_OVER_LEFT  
LV_SCR_LOAD_ANIM_OVER_RIGHT  
LV_SCR_LOAD_ANIM_OVER_TOP  
LV_SCR_LOAD_ANIM_OVER_BOTTOM  
LV_SCR_LOAD_ANIM_MOVE_LEFT  
LV_SCR_LOAD_ANIM_MOVE_RIGHT  
LV_SCR_LOAD_ANIM_MOVE_TOP  
LV_SCR_LOAD_ANIM_MOVE_BOTTOM  
LV_SCR_LOAD_ANIM_FADE_IN  
LV_SCR_LOAD_ANIM_FADE_ON  
LV_SCR_LOAD_ANIM_FADE_OUT  
LV_SCR_LOAD_ANIM_OUT_LEFT  
LV_SCR_LOAD_ANIM_OUT_RIGHT  
LV_SCR_LOAD_ANIM_OUT_TOP
```

**LV\_SCR\_LOAD\_ANIM\_OUT\_BOTTOM**

**lv\_ft81x\_spi\_operation**

## 14.5 lv\_conf\_internal.h

GENERATED FILE, DO NOT EDIT IT!

This file ensures all defines of lv\_conf.h have a default value.

### Defines

**LV\_OS\_NONE**

**LV\_OS\_PTHREAD**

**LV\_OS\_FREERTOS**

**LV\_OS\_CMSIS\_RTOS2**

**LV\_OS\_RTTHREAD**

**LV\_OS\_WINDOWS**

**LV\_OS\_MQX**

**LV\_OS SDL2**

**LV\_OS\_CUSTOM**

**LV\_STDLIB\_BUILTIN**

**LV\_STDLIB\_CLIB**

**LV\_STDLIB\_MICROPYTHON**

**LV\_STDLIB\_RTTHREAD**

**LV\_STDLIB\_CUSTOM**

**LV\_DRAW\_SW\_ASM\_NONE**

**LV\_DRAW\_SW\_ASM\_NEON**

**LV\_DRAW\_SW\_ASM\_HELIUM**

**LV\_DRAW\_SW\_ASM\_CUSTOM****LV\_NEMA\_HAL\_CUSTOM****LV\_NEMA\_HAL\_STM32****LV\_COLOR\_DEPTH**

Handle special Kconfig options. Color depth: 1 (I1), 8 (L8), 16 (RGB565), 24 (RGB888), 32 (XRGB8888)

**LV\_USE\_STDLIB\_MALLOC**

Possible values

- LV\_STDLIB\_BUILTIN: LVGL's built in implementation
- LV\_STDLIB\_CLIB: Standard C functions, like malloc, strlen, etc
- LV\_STDLIB\_MICROPYTHON: MicroPython implementation
- LV\_STDLIB\_RTTHREAD: RT-Thread implementation
- LV\_STDLIB\_CUSTOM: Implement the functions externally

**LV\_USE\_STDLIB\_STRING**

Possible values

- LV\_STDLIB\_BUILTIN: LVGL's built in implementation
- LV\_STDLIB\_CLIB: Standard C functions, like malloc, strlen, etc
- LV\_STDLIB\_MICROPYTHON: MicroPython implementation
- LV\_STDLIB\_RTTHREAD: RT-Thread implementation
- LV\_STDLIB\_CUSTOM: Implement the functions externally

**LV\_USE\_STDLIB\_SPRINTF**

Possible values

- LV\_STDLIB\_BUILTIN: LVGL's built in implementation
- LV\_STDLIB\_CLIB: Standard C functions, like malloc, strlen, etc
- LV\_STDLIB\_MICROPYTHON: MicroPython implementation
- LV\_STDLIB\_RTTHREAD: RT-Thread implementation
- LV\_STDLIB\_CUSTOM: Implement the functions externally

**LV\_STDINT\_INCLUDE****LV\_STDDEF\_INCLUDE****LV\_STDBOOL\_INCLUDE****LV\_INTTYPES\_INCLUDE****LV\_LIMITS\_INCLUDE****LV\_STDARG\_INCLUDE****LV\_MEM\_SIZE**

Size of memory available for `lv_malloc()` in bytes (>= 2kB) [bytes]

**LV\_MEM\_POOL\_EXPAND\_SIZE**

Size of the memory expand for `lv_malloc()` in bytes

**LV\_MEM\_ADDR**

Set an address for the memory pool instead of allocating it as a normal array. Can be in external SRAM too.  
0: unused

**LV\_DEF\_REFR\_PERIOD**

Default display refresh, input device read and animation step period. [ms]

**LV\_DPI\_DEF**

Default Dots Per Inch. Used to initialize default sizes such as widgets sized, style paddings. (Not so important, you can adjust it to modify default sizes and spaces.) [px/inch]

**LV\_USE\_OS**

Select operating system to use. Possible options:

- LV\_OS\_NONE
- LV\_OS\_PTHREAD
- LV\_OS\_FREERTOS
- LV\_OS\_CMSIS\_RTOS2
- LV\_OS\_RTTHREAD
- LV\_OS\_WINDOWS
- LV\_OS\_MQX
- LV\_OS\_SDL2
- LV\_OS\_CUSTOM

**LV\_DRAW\_BUF\_STRIDE\_ALIGN**

Align stride of all layers and images to this bytes

**LV\_DRAW\_BUF\_ALIGN**

Align start address of draw\_buf addresses to this bytes

**LV\_DRAW\_TRANSFORM\_USE\_MATRIX**

Using matrix for transformations. Requirements:

- LV\_USE\_MATRIX = 1.
- Rendering engine needs to support 3x3 matrix transformations.

**LV\_DRAW\_LAYER\_SIMPLE\_BUF\_SIZE**

The target buffer size for simple layer chunks. [bytes]

**LV\_DRAW\_LAYER\_MAX\_MEMORY**

No limit by default [bytes]

**LV\_DRAW\_THREAD\_STACK\_SIZE**

Stack size of drawing thread. NOTE: If FreeType or ThorVG is enabled, it is recommended to set it to 32KB or more. [bytes]

**LV\_DRAW\_THREAD\_PRIO**

Thread priority of the drawing task. Higher values mean higher priority. Can use values from lv\_thread\_prio\_t enum in lv\_os.h: LV\_THREAD\_PRIO\_LOWEST, LV\_THREAD\_PRIO\_LOW, LV\_THREAD\_PRIO\_MID, LV\_THREAD\_PRIO\_HIGH, LV\_THREAD\_PRIO\_HIGHEST Make sure the priority value aligns with the OS-specific priority levels. On systems with limited priority levels (e.g., FreeRTOS), a higher value can improve rendering performance but might cause other tasks to starve.

**LV\_USE\_DRAW\_SW**

**LV\_DRAW\_SW\_SUPPORT\_RGB565**

**LV\_DRAW\_SW\_SUPPORT\_RGB565\_SWAPPED**

**LV\_DRAW\_SW\_SUPPORT\_RGB565A8**

**LV\_DRAW\_SW\_SUPPORT\_RGB888**

**LV\_DRAW\_SW\_SUPPORT\_XRGB8888**

**LV\_DRAW\_SW\_SUPPORT\_ARGB8888**

**LV\_DRAW\_SW\_SUPPORT\_ARGB8888\_PREMULTIPLIED**

**LV\_DRAW\_SW\_SUPPORT\_L8**

**LV\_DRAW\_SW\_SUPPORT\_AL88**

**LV\_DRAW\_SW\_SUPPORT\_A8**

**LV\_DRAW\_SW\_SUPPORT\_I1**

**LV\_DRAW\_SW\_I1\_LUM\_THRESHOLD**

**LV\_DRAW\_SW\_DRAW\_UNIT\_CNT**

Set number of draw units.

- > 1 requires operating system to be enabled in **LV\_USE\_OS**.
- > 1 means multiple threads will render the screen in parallel.

**LV\_USE\_DRAW\_ARM2D\_SYNC**

Use Arm-2D to accelerate software (sw) rendering.

**LV\_USE\_NATIVE\_HELIUM\_ASM**

Enable native helium assembly to be compiled.

**LV\_DRAW\_SW\_COMPLEX**

- 0: Use a simple renderer capable of drawing only simple rectangles with gradient, images, text, and straight lines only.
- 1: Use a complex renderer capable of drawing rounded corners, shadow, skew lines, and arcs too.

**LV\_DRAW\_SW\_SHADOW\_CACHE\_SIZE**

Allow buffering some shadow calculation. **LV\_DRAW\_SW\_SHADOW\_CACHE\_SIZE** is the maximum shadow size to buffer, where shadow size is `shadow_width + radius`. Caching has `LV_DRAW_SW_SHADOW_CACHE_SIZE^2` RAM cost.

**LV\_DRAW\_SW\_CIRCLE\_CACHE\_SIZE**

Set number of maximally-cached circle data. The circumference of 1/4 circle are saved for anti-aliasing.  
radius \* 4 bytes are used per circle (the most often used radii are saved).

- 0: disables caching

**LV\_USE\_DRAW\_SW\_ASM****LV\_USE\_DRAW\_SW\_COMPLEX\_GRADIENTS**

Enable drawing complex gradients in software: linear at an angle, radial or conical

**LV\_USE\_NEMA\_GFX****LV\_USE\_DRAW\_VGLITE**

Use NXP's VG-Lite GPU on iMX RTxxx platforms.

**LV\_USE\_PXP**

Use NXP's PXP on iMX RTxxx platforms.

**LV\_USE\_DRAW\_G2D**

Use NXP's G2D on MPU platforms.

**LV\_USE\_DRAW\_DAVE2D**

Use Renesas Dave2D on RA platforms.

**LV\_USE\_DRAW SDL**

Draw using cached SDL textures

**LV\_USE\_DRAW\_VG\_LITE**

Use VG-Lite GPU.

**LV\_USE\_DRAW\_DMA2D**

Accelerate blends, fills, etc. with STM32 DMA2D

**LV\_USE\_DRAW\_OPENGLS**

Draw using cached OpenGL ES textures

**LV\_USE\_PPA**

Draw using espressif PPA accelerator

**LV\_USE\_DRAW\_EVE****LV\_USE\_LOG**

Enable log module

**LV\_USE\_ASSERT\_NULL**

Check if the parameter is NULL. (Very fast, recommended)

**LV\_USE\_ASSERT\_MALLOC**

Checks if the memory is successfully allocated or not. (Very fast, recommended)

**LV\_USE\_ASSERT\_STYLE**

Check if the styles are properly initialized. (Very fast, recommended)

**LV\_USE\_ASSERT\_MEM\_INTEGRITY**

Check the integrity of `lv_mem` after critical operations. (Slow)

**LV\_USE\_ASSERT\_OBJ**

Check the object's type and existence (e.g. not deleted). (Slow)

**LV\_ASSERT\_HANDLER\_INCLUDE**

Add a custom handler when assert happens e.g. to restart MCU.

**LV\_ASSERT\_HANDLER**

Halt by default

**LV\_USE\_REFR\_DEBUG**

1: Draw random colored rectangles over the redrawn areas.

**LV\_USE\_LAYER\_DEBUG**

1: Draw a red overlay for ARGB layers and a green overlay for RGB layers

**LV\_USE\_PARALLEL\_DRAW\_DEBUG**

1: Adds the following behaviors for debugging:

- Draw overlays with different colors for each draw\_unit's tasks.
- Draw index number of draw unit on white background.
- For layers, draws index number of draw unit on black background.

**LV\_ENABLE\_GLOBAL\_CUSTOM****LV\_CACHE\_DEF\_SIZE**

Default cache size in bytes. Used by image decoders such as `lv_lodepng` to keep the decoded image in memory. If size is not set to 0, the decoder will fail to decode when the cache is full. If size is 0, the cache function is not enabled and the decoded memory will be released immediately after use.

**LV\_IMAGE\_HEADER\_CACHE\_DEF\_CNT**

Default number of image header cache entries. The cache is used to store the headers of images. The main logic is like `LV_CACHE_DEF_SIZE` but for image headers.

**LV\_GRADIENT\_MAX\_STOPS**

Number of stops allowed per gradient. Increase this to allow more stops. This adds `(sizeof(lv_color_t) + 1)` bytes per additional stop.

**LV\_COLOR\_MIX\_ROUND\_OFS**

Adjust color mix functions rounding. GPUs might calculate color mix (blending) differently.

- 0: round down,
- 64: round up from x.75,
- 128: round up from half,
- 192: round up from x.25,
- 254: round up

**LV\_OBJ\_STYLE\_CACHE**

Add 2 x 32-bit variables to each `lv_obj_t` to speed up getting style properties

**LV\_USE\_OBJ\_ID**

Add `id` field to `lv_obj_t`

**LV\_USE\_OBJ\_NAME**

Enable support widget names

**LV\_OBJ\_ID\_AUTO\_ASSIGN**

Automatically assign an ID when obj is created

**LV\_USE\_OBJ\_ID\_BUILTIN**

Use builtin obj ID handler functions:

- `lv_obj_assign_id`: Called when a widget is created. Use a separate counter for each widget class as an ID.
- `lv_obj_id_compare`: Compare the ID to decide if it matches with a requested value.
- `lv_obj_stringify_id`: Return string-ified identifier, e.g. "button3".
- `lv_obj_free_id`: Does nothing, as there is no memory allocation for the ID. When disabled these functions needs to be implemented by the user.

**LV\_USE\_OBJ\_PROPERTY**

Use obj property set/get API.

**LV\_USE\_OBJ\_PROPERTY\_NAME**

Enable property name support.

**LV\_USE\_VG\_LITE\_THORVG****LV\_USE\_GESTURE\_RECOGNITION****LV\_BIG\_ENDIAN\_SYSTEM**

For big endian systems set to 1

**LV\_ATTRIBUTE\_MEM\_ALIGN\_SIZE**

Define a custom attribute for `lv_tick_inc` function Define a custom attribute for `lv_timer_handler` function Define a custom attribute for `lv_display_flush_ready` function Align VG\_LITE buffers on this number of bytes.

**Nota**

`vglite_src_buf_aligned()` uses this value to validate alignment of passed buffer pointers.

**LV\_EXPORT\_CONST\_INT(int\_value)**

Will be added where memory needs to be aligned (with -Os data might not be aligned to boundary by default). E.g. `attribute((aligned(4)))` Attribute to mark large constant arrays, for example for font bitmaps Compiler prefix for a large array declaration in RAM Place performance critical functions into a faster memory (e.g RAM) Export integer constant to binding. This macro is used with constants in the form of `LV_<CONST>` that should also appear on LVGL binding API such as MicroPython. The default value just prevents GCC warning

**LV\_USE\_FLOAT**

Prefix all global extern data with this Use float as lv\_value\_precise\_t

**LV\_USE\_MATRIX**

Enable matrix support

- Requires LV\_USE\_FLOAT = 1

**LV\_USE\_PRIVATE\_API**

Include lvgl\_private.h in lvgl.h to access internal data and functions by default

**LV\_FONT\_MONTserrat\_8****LV\_FONT\_MONTserrat\_10****LV\_FONT\_MONTserrat\_12****LV\_FONT\_MONTserrat\_14****LV\_FONT\_MONTserrat\_16****LV\_FONT\_MONTserrat\_18****LV\_FONT\_MONTserrat\_20****LV\_FONT\_MONTserrat\_22****LV\_FONT\_MONTserrat\_24****LV\_FONT\_MONTserrat\_26****LV\_FONT\_MONTserrat\_28****LV\_FONT\_MONTserrat\_30****LV\_FONT\_MONTserrat\_32****LV\_FONT\_MONTserrat\_34****LV\_FONT\_MONTserrat\_36****LV\_FONT\_MONTserrat\_38****LV\_FONT\_MONTserrat\_40****LV\_FONT\_MONTserrat\_42**

**LV\_FONT\_MONTserrat\_44****LV\_FONT\_MONTserrat\_46****LV\_FONT\_MONTserrat\_48****LV\_FONT\_MONTserrat\_28\_COMPRESSED**

bpp = 3

**LV\_FONT\_DEJAVU\_16\_PERSIAN\_HEBREW**

Hebrew, Arabic, Persian letters and all their forms

**LV\_FONT\_SOURCE\_HAN\_SANS\_SC\_14\_CJK**

1338 most common CJK radicals

**LV\_FONT\_SOURCE\_HAN\_SANS\_SC\_16\_CJK**

1338 most common CJK radicals

**LV\_FONT\_UNSCII\_8**

Pixel perfect monospaced fonts

**LV\_FONT\_UNSCII\_16****LV\_FONT\_CUSTOM\_DECLARE**

Optionally declare custom fonts here.

You can use any of these fonts as the default font too and they will be available globally. Example:

```
#define LV_FONT_CUSTOM_DECLARE    LV_FONT_DECLARE(my_font_1) LV_FONT_DECLARE(my_font_2)
```

**LV\_FONT\_DEFAULT**

Always set a default font

**LV\_FONT\_FMT\_TXT\_LARGE**

Enable handling large font and/or fonts with a lot of characters. The limit depends on the font size, font face and bpp. A compiler error will be triggered if a font needs it.

**LV\_USE\_FONT\_COMPRESSED**

Enables/disables support for compressed fonts.

**LV\_USE\_FONT\_PLACEHOLDER**

Enable drawing placeholders when glyph dsc is not found.

**LV\_TXT\_ENC**

Select a character encoding for strings. Your IDE or editor should have the same character encoding.

- LV\_TXT\_ENC\_UTF8
- LV\_TXT\_ENC\_ASCII

**LV\_TXT\_BREAK\_CHARS**

While rendering text strings, break (wrap) text on these chars.

**LV\_TXT\_LINE\_BREAK\_LONG\_LEN**

If a word is at least this long, will break wherever "prettiest". To disable, set to a value  $\leq 0$ .

**LV\_TXT\_LINE\_BREAK\_LONG\_PRE\_MIN\_LEN**

Minimum number of characters in a long word to put on a line before a break. Depends on LV\_TXT\_LINE\_BREAK\_LONG\_LEN.

**LV\_TXT\_LINE\_BREAK\_LONG\_POST\_MIN\_LEN**

Minimum number of characters in a long word to put on a line after a break. Depends on LV\_TXT\_LINE\_BREAK\_LONG\_LEN.

**LV\_USE\_BIDI**

Support bidirectional text. Allows mixing Left-to-Right and Right-to-Left text. The direction will be processed according to the Unicode Bidirectional Algorithm: <https://www.w3.org/International/articles/inline-bidi-markup/uba-basics>

**LV\_USE\_ARABIC\_PERSIAN\_CHARS**

Enable Arabic/Persian processing In these languages characters should be replaced with another form based on their position in the text

**LV\_TXT\_COLOR\_CMD****LV\_WIDGETS\_HAS\_DEFAULT\_VALUE**

1: Causes these widgets to be given default values at creation time.

- lv\_buttonmatrix\_t: Get default maps: {"Btn1", "Btn2", "Btn3", "\n", "Btn4", "Btn5", ""}, else map not set.
- lv\_checkbox\_t : String label set to "Check box", else set to empty string.
- lv\_dropdown\_t : Options set to "Option 1", "Option 2", "Option 3", else no values are set.
- lv\_roller\_t : Options set to "Option 1", "Option 2", "Option 3", "Option 4", "Option 5", else no values are set.
- lv\_label\_t : Text set to "Text", else empty string.
- lv\_arclabel\_t : Text set to "Arced Text", else empty string.

**LV\_USE\_ANIMIMG****LV\_USE\_ARC****LV\_USE\_ARCLABEL****LV\_USE\_BAR****LV\_USE\_BUTTON****LV\_USE\_BUTTONMATRIX****LV\_USE\_CALENDAR****LV\_CALENDAR\_WEEK\_STARTS\_MONDAY**

**LV\_CALENDAR\_DEFAULT\_DAY\_NAMES**

**LV\_CALENDAR\_DEFAULT\_MONTH\_NAMES**

**LV\_USE\_CALENDAR\_HEADER\_ARROW**

**LV\_USE\_CALENDAR\_HEADER\_DROPDOWN**

**LV\_USE\_CALENDAR\_CHINESE**

**LV\_USE\_CANVAS**

**LV\_USE\_CHART**

**LV\_USE\_CHECKBOX**

**LV\_USE\_DROPDOWN**

Requires: lv\_label

**LV\_USE\_IMAGE**

Requires: lv\_label

**LV\_USE\_IMAGEBUTTON**

**LV\_USE\_KEYBOARD**

**LV\_USE\_LABEL**

**LV\_LABEL\_TEXT\_SELECTION**

Enable selecting text of the label

**LV\_LABEL\_LONG\_TXT\_HINT**

Store some extra info in labels to speed up drawing of very long text

**LV\_LABEL\_WAIT\_CHAR\_COUNT**

The count of wait chart

**LV\_USE\_LED**

**LV\_USE\_LINE**

**LV\_USE\_LIST**

**LV\_USE\_LOTTIE**

Requires: lv\_canvas, thorvg

**LV\_USE\_MENU**

**LV\_USE\_MSGBOX**

**LV\_USE\_ROLLER**

Requires: lv\_label

**LV\_USE\_SCALE**

**LV\_USE\_SLIDER**

Requires: lv\_bar

**LV\_USE\_SPAN**

**LV\_SPAN\_SNIPPET\_STACK\_SIZE**

A line of text can contain this maximum number of span descriptors.

**LV\_USE\_SPINBOX**

**LV\_USE\_SPINNER**

**LV\_USE\_SWITCH**

**LV\_USE\_TABLE**

**LV\_USE\_TABVIEW**

**LV\_USE\_TEXTAREA**

Requires: lv\_label

**LV\_TEXTAREA\_DEF\_PWD\_SHOW\_TIME**

[ms]

**LV\_USE\_TILEVIEW**

**LV\_USE\_WIN**

**LV\_USE\_3DTEXTURE**

**LV\_USE\_THEME\_DEFAULT**

A simple, impressive and very complete theme

**LV\_THEME\_DEFAULT\_DARK**

0: Light mode; 1: Dark mode

**LV\_THEME\_DEFAULT\_GROW**

1: Enable grow on press

**LV\_THEME\_DEFAULT\_TRANSITION\_TIME**

Default transition time in ms.

**LV\_USE\_THEME\_SIMPLE**

A very simple theme that is a good starting point for a custom theme

**LV\_USE\_THEME\_MONO**

A theme designed for monochrome displays

**LV\_USE\_FLEX**

A layout similar to Flexbox in CSS.

**LV\_USE\_GRID**

A layout similar to Grid in CSS.

**LV\_FS\_DEFAULT\_DRIVER LETTER**

Setting a default driver letter allows skipping the driver prefix in filepaths. Documentation about how to use the below driver-identifier letters can be found at <https://docs.lvgl.io/master/details/main-modules/fs.html#lv-fs-identifier-letters>.

**LV\_USE\_FS\_STDIO**

API for fopen, fread, etc.

**LV\_USE\_FS\_POSIX**

API for open, read, etc.

**LV\_USE\_FS\_WIN32**

API for CreateFile, ReadFile, etc.

**LV\_USE\_FS\_FATFS**

API for FATFS (needs to be added separately). Uses f\_open, f\_read, etc.

**LV\_USE\_FS\_MEMFS**

API for memory-mapped file access.

**LV\_USE\_FS\_LITTLEFS**

API for LittleFs.

**LV\_USE\_FS\_ARDUINO\_ESP\_LITTLEFS**

API for Arduino LittleFs.

**LV\_USE\_FS\_ARDUINO\_SD**

API for Arduino Sd.

**LV\_USE\_FS\_UEFI**

API for UEFI

**LV\_USE\_LODEPNG**

LODEPNG decoder library

**LV\_USE\_LIBPNG**

PNG decoder(libpng) library

**LV\_USE\_BMP**

BMP decoder library

**LV\_USE\_TJPGD**

JPG + split JPG decoder library. Split JPG is a custom format optimized for embedded systems.

**LV\_USE\_LIBJPEG\_TURBO**

libjpeg-turbo decoder library.

- Supports complete JPEG specifications and high-performance JPEG decoding.

**LV\_USE\_GIF**

GIF decoder library

**LV\_BIN\_DECODER\_RAM\_LOAD**

Decode bin images to RAM

**LV\_USE\_RLE**

RLE decompress library

**LV\_USE\_QRCODE**

QR code library

**LV\_USE\_BARCODE**

Barcode code library

**LV\_USE\_FREETYPE**

FreeType library

**LV\_USE\_TINY\_TTF**

Built-in TTF decoder

**LV\_USE\_RLOTTIE**

Rlottie library

**LV\_USE\_VECTOR\_GRAPHIC**

Enable Vector Graphic APIs

- Requires `LV_USE_MATRIX = 1`

**LV\_USE\_THORVG\_INTERNAL**

Enable ThorVG (vector graphics library) from the src/libs folder

**LV\_USE\_THORVG\_EXTERNAL**

Enable ThorVG by assuming that its installed and linked to the project

**LV\_USE\_LZ4\_INTERNAL**

Use lvgl built-in LZ4 lib

**LV\_USE\_LZ4\_EXTERNAL**

Use external LZ4 library

**LV\_USE\_SVG****LV\_USE\_SVG\_ANIMATION****LV\_USE\_SVG\_DEBUG****LV\_USE\_FFMPEG**

FFmpeg library for image decoding and playing videos. Supports all major image formats so do not enable other image decoder with it.

**LV\_USE\_SNAPSHOT**

1: Enable API to take snapshot for object

**LV\_USE\_SYSMON**

1: Enable system monitor component

**LV\_USE\_PROFILER**

1: Enable runtime performance profiler

**LV\_USE\_MONKEY**

1: Enable Monkey test

**LV\_USE\_GRIDNAV**

1: Enable grid navigation

**LV\_USE\_FRAMENT**

1: Enable lv\_obj fragment logic

**LV\_USE\_IMGFONT**

1: Support using images as font in label or span widgets

**LV\_USE\_OBSERVER**

1: Enable an observer pattern implementation

**LV\_USE\_IME\_PINYIN**

1: Enable Pinyin input method

- Requires: lv\_keyboard

**LV\_USE\_FILE\_EXPLORER**

1: Enable file explorer.

- Requires: lv\_table

**LV\_USE\_FONT\_MANAGER**

1: Enable Font manager

## **LV\_USE\_TEST**

Enable emulated input devices, time emulation, and screenshot compares.

## **LV\_USE\_XML**

Enable loading XML UIs runtime

## **LV\_USE\_TRANSLATION**

1: Enable text translation support

## **LV\_USE\_COLOR\_FILTER**

## **LV\_USE SDL**

Use SDL to open window on PC and handle mouse and keyboard.

## **LV\_USE\_X11**

Use X11 to open window on Linux desktop and handle mouse and keyboard

## **LV\_USE\_WAYLAND**

Use Wayland to open a window and handle input on Linux or BSD desktops

## **LV\_USE\_LINUX\_FBDEV**

Driver for /dev/fb

## **LV\_USE\_NUTTX**

Use Nuttx to open window and handle touchscreen

## **LV\_USE\_LINUX\_DRM**

Driver for /dev/dri/card

## **LV\_USE\_TFT\_ESPI**

Interface for TFT\_eSPI

## **LV\_USE\_EVDEV**

Driver for evdev input devices

## **LV\_USE\_LIBINPUT**

Driver for libinput input devices

## **LV\_USE\_ST7735**

## **LV\_USE\_ST7789**

## **LV\_USE\_ST7796**

## **LV\_USE\_ILI9341**

## **LV\_USE\_FT81X**

**LV\_USE\_GENERIC\_MIPI****LV\_USE\_RENESAS\_GLCD**

Driver for Renesas GLCD

**LV\_USE\_ST\_LTDC**

Driver for ST LTDC

**LV\_USE\_NXP\_ELCDIF**

Driver for NXP ELCDIF

**LV\_USE\_WINDOWS**

LVGL Windows backend

**LV\_USE\_UEFI**

LVGL UEFI backend

**LV\_USE\_OPENGLS**

Use OpenGL to open window on PC and handle mouse and keyboard

**LV\_USE\_QNX**

QNX Screen display and input drivers

**LV\_BUILD\_EXAMPLES**

Enable examples to be built with the library.

**LV\_BUILD\_DEMOS**

Build the demos

**LV\_USE\_DEMO\_WIDGETS**

Show some widgets. This might be required to increase LV\_MEM\_SIZE.

**LV\_USE\_DEMO\_KEYPAD\_AND\_ENCODER**

Demonstrate usage of encoder and keyboard.

**LV\_USE\_DEMO\_BENCHMARK**

Benchmark your system

**LV\_USE\_DEMO\_RENDER**

Render test for each primitive.

- Requires at least 480x272 display.

**LV\_USE\_DEMO\_STRESS**

Stress test for LVGL

**LV\_USE\_DEMO\_MUSIC**

Music player demo

**LV\_USE\_DEMO\_VECTOR\_GRAPHIC**

Vector graphic demo

**LV\_USE\_DEMO\_FLEX\_LAYOUT**

Flex layout demo

**LV\_USE\_DEMO\_MULTILANG**

Smart-phone like multi-language demo

**LV\_USE\_DEMO\_TRANSFORM**

Widget transformation demo

**LV\_USE\_DEMO\_SCROLL**

Demonstrate scroll settings

**LV\_USE\_DEMO\_EBIKE****LV\_USE\_DEMO\_HIGH\_RES**

High-resolution demo

**LV\_USE\_DEMO\_SMARTWATCH****LV\_USE\_ANIMIMAGE****LV\_LOG\_LEVEL****LV\_LOG\_TRACE\_MEM****LV\_LOG\_TRACE\_TIMER****LV\_LOG\_TRACE\_INDEV****LV\_LOG\_TRACE\_DISP\_REFR****LV\_LOG\_TRACE\_EVENT****LV\_LOG\_TRACE\_OBJ\_CREATE****LV\_LOG\_TRACE\_LAYOUT****LV\_LOG\_TRACE\_ANIM****LV\_WAYLAND\_USE\_DMABUF****LV\_WAYLAND\_WINDOW\_DECORATIONS**

**LV\_WAYLAND\_WL\_SHELL****LV\_USE\_PERF\_MONITOR****LV\_USE\_MEM\_MONITOR****LV\_USE\_PERF\_MONITOR\_LOG\_MODE****LV\_USE\_LZ4****LV\_USE\_THORVG****LV\_FS\_IS\_VALID\_LETTER(1)**

## 14.6 lv\_conf\_kconfig.h

- \* Configs that need special handling when LVGL is used with Kconfig

## 14.7 lv\_init.h

### Functions

**void lv\_init(void)**

Initialize LVGL library. Should be called before any other LVGL related function.

**void lv\_deinit(void)**

Deinit the 'lv' library

**bool lv\_is\_initialized(void)**

Returns whether the 'lv' library is currently initialized

## 14.8 lvgl.h

This file exists only to be compatible with Arduino's library structure

## 14.9 lvgl\_private.h

This file exists only to be compatible with Arduino's library structure

## 14.10 core

### 14.10.1 lv\_global.h

#### Defines

**ZERO\_MEM\_SENTINEL**

**LV\_GLOBAL\_DEFAULT()**

**Typedef**

```
typedef struct _lv_global_t lv_global_t
```

**Variables**

*lv\_global\_t* **lv\_global**

```
struct lv_global_t
```

**Public Members**

bool **inited**

bool **deinit\_in\_progress**

Can be used e.g. in the LV\_EVENT\_DELETE to deinit the drivers too

*lv\_ll\_t* **disp\_ll**

lv\_display\_t \***disp\_refresh**

lv\_display\_t \***disp\_default**

*lv\_ll\_t* **style\_trans\_ll**

bool **style\_refresh**

uint32\_t **style\_custom\_table\_size**

uint32\_t **style\_last\_custom\_prop\_id**

uint8\_t \***style\_custom\_prop\_flag\_lookup\_table**

*lv\_ll\_t* **group\_ll**

lv\_group\_t \***group\_default**

*lv\_ll\_t* **indev\_ll**

lv\_indev\_t \***indev\_active**

lv\_obj\_t \***indev\_obj\_active**

uint32\_t **layout\_count**

*lv\_layout\_dsc\_t* \***layout\_list**

```
bool layout_update_mutex

uint32_t memory_zero

uint32_t math_rand_seed

lv_event_t *event_header

uint32_t event_last_register_id

lv_timer_state_t timer_state

lv_anim_state_t anim_state

lv_tick_state_t tick_state

lv_draw_buf_handlers_t draw_buf_handlers

lv_draw_buf_handlers_t font_draw_buf_handlers

lv_draw_buf_handlers_t image_cache_draw_buf_handlers
    Ensure that all assigned draw buffers can be managed by image cache.

lv_ll_t img_decoder_ll

lv_cache_t *img_cache

lv_cache_t *img_header_cache

lv_draw_global_info_t draw_info

lv_ll_t draw_sw_blend_handler_ll

lv_draw_sw_mask_radius_circle_dsc_arr_t sw_circle_cache

lv_log_print_g_cb_t custom_log_print_cb

uint32_t log_last_log_time

void *theme_simple

void *theme_default

void *theme_mono
```

```
lv_tlsf_state_t tlsf_state  
  
lv_ll_t fsdrv_ll  
  
lv_fs_drv_t stdio_fs_drv  
  
lv_fs_drv_t posix_fs_drv  
  
lv_fs_drv_t fatfs_fs_drv  
  
lv_fs_drv_t win32_fs_drv  
  
lv_fs_drv_t uefi_fs_drv  
  
lv_fs_drv_t arduino_esp_littlefs_fs_drv  
  
lv_fs_drv_t arduino_sd_fs_drv  
  
struct _lv_freetype_context_t *ft_context  
  
lv_font_fmt_rle_t font_fmt_rle  
  
struct _snippet_stack *span_snippet_stack  
  
lv_style_t fe_list_button_style  
  
lv_sysmon_backend_data_t sysmon_mem  
  
size_t ime_cand_len  
  
void *objid_array  
  
uint32_t objid_count  
  
lv_test_state_t test_state  
  
lv_ll_t translation_packs_ll  
  
const char *translation_selected_lang  
  
struct _lv_nuttx_ctx_t *nuttx_ctx  
  
lv_evdev_discovery_t *evdev_discovery
```

```
const char *xml_path_prefix  
  
uint32_t lv_event_xml_store_timeline  
  
lv_draw_eve_unit_t *draw_eve_unit  
  
void *user_data
```

## 14.10.2 lv\_group.h

### Typedef

```
typedef void (*lv_group_focus_cb_t)(lv_group_t*)
```

```
typedef void (*lv_group_edge_cb_t)(lv_group_t*, bool)
```

### Enums

#### enum **lv\_key\_t**

Predefined keys to control which Widget has focus via lv\_group\_send(group, c)

*Values:*

enumerator **LV\_KEY\_UP**

enumerator **LV\_KEY\_DOWN**

enumerator **LV\_KEY\_RIGHT**

enumerator **LV\_KEY\_LEFT**

enumerator **LV\_KEY\_ESC**

enumerator **LV\_KEY\_DEL**

enumerator **LV\_KEY\_BACKSPACE**

enumerator **LV\_KEY\_ENTER**

enumerator **LV\_KEY\_NEXT**

enumerator **LV\_KEY\_PREV**

enumerator **LV\_KEY\_HOME**

enumerator **LV\_KEY\_END**

---

enum **lv\_group\_refocus\_policy\_t**

*Values:*

enumerator **LV\_GROUP\_REFOCUS\_POLICY\_NEXT**

enumerator **LV\_GROUP\_REFOCUS\_POLICY\_PREV**

## Functions

**lv\_group\_t \*lv\_group\_create(void)**

Create new Widget group.

**Ritorna**

pointer to the new Widget group

**void lv\_group\_delete(lv\_group\_t \*group)**

Delete group object.

**Parametri**

**group** -- pointer to a group

**void lv\_group\_set\_default(lv\_group\_t \*group)**

Set default group. New Widgets will be added to this group if it's enabled in their class with `add_to_def_group = true`.

**Parametri**

**group** -- pointer to a group (can be NULL)

**lv\_group\_t \*lv\_group\_get\_default(void)**

Get default group.

**Ritorna**

pointer to the default group

**void lv\_group\_add\_obj(lv\_group\_t \*group, lv\_obj\_t \*obj)**

Add an Widget to group.

**Parametri**

- **group** -- pointer to a group
- **obj** -- pointer to a Widget to add

**void lv\_group\_swap\_obj(lv\_obj\_t \*obj1, lv\_obj\_t \*obj2)**

Swap 2 Widgets in group. Widgets must be in the same group.

**Parametri**

- **obj1** -- pointer to a Widget
- **obj2** -- pointer to another Widget

**void lv\_group\_remove\_obj(lv\_obj\_t \*obj)**

Remove a Widget from its group.

**Parametri**

**obj** -- pointer to Widget to remove

**void lv\_group\_remove\_all\_objs(lv\_group\_t \*group)**

Remove all Widgets from a group.

**Parametri**

**group** -- pointer to a group

**void lv\_group\_focus\_obj(lv\_obj\_t \*obj)**

Focus on a Widget (defocus the current).

**Parametri**

**obj** -- pointer to Widget to focus on

**void lv\_group\_focus\_next(lv\_group\_t \*group)**

Focus on next Widget in a group (defocus the current).

**Parametri**

**group** -- pointer to a group

**void lv\_group\_focus\_prev(lv\_group\_t \*group)**

Focus on previous Widget in a group (defocus the current).

**Parametri**

**group** -- pointer to a group

**void lv\_group\_focus\_freeze(lv\_group\_t \*group, bool en)**

Do not allow changing focus from current Widget.

**Parametri**

- **group** -- pointer to a group
- **en** -- true: freeze, false: release freezing (normal mode)

**lv\_result\_t lv\_group\_send\_data(lv\_group\_t \*group, uint32\_t c)**

Send a control character to Widget that has focus in a group.

**Parametri**

- **group** -- pointer to a group
- **c** -- a character (use LV\_KEY\_.. to navigate)

**Ritorna**

result of Widget with focus in group.

**void lv\_group\_set\_focus\_cb(lv\_group\_t \*group, lv\_group\_focus\_cb\_t focus\_cb)**

Set a function for a group which will be called when a new Widget has focus.

**Parametri**

- **group** -- pointer to a group
- **focus\_cb** -- the call back function or NULL if unused

**void lv\_group\_set\_edge\_cb(lv\_group\_t \*group, lv\_group\_edge\_cb\_t edge\_cb)**

Set a function for a group which will be called when a focus edge is reached

**Parametri**

- **group** -- pointer to a group
- **edge\_cb** -- the call back function or NULL if unused

**void lv\_group\_set\_refocus\_policy(lv\_group\_t \*group, lv\_group\_refocus\_policy\_t policy)**

Set whether the next or previous Widget in a group gets focus when Widget that has focus is deleted.

**Parametri**

- **group** -- pointer to a group
- **policy** -- new refocus policy enum

**void lv\_group\_set\_editing(lv\_group\_t \*group, bool edit)**

Manually set the current mode (edit or navigate).

**Parametri**

- **group** -- pointer to group
- **edit** -- true: edit mode; false: navigate mode

**void lv\_group\_set\_wrap(lv\_group\_t \*group, bool en)**

Set whether moving focus to next/previous Widget will allow wrapping from first->last or last->first Widget.

**Parametri**

- **group** -- pointer to group
- **en** -- true: wrapping enabled; false: wrapping disabled

**lv\_obj\_t \*lv\_group\_get\_focused(const lv\_group\_t \*group)**

Get Widget that has focus, or NULL if there isn't one.

**Parametri**

**group** -- pointer to a group

**Ritorna**

pointer to Widget with focus

*lv\_group\_focus\_cb\_t* **lv\_group\_get\_focus\_cb**(const lv\_group\_t \*group)

Get focus callback function of a group.

**Parametri**

**group** -- pointer to a group

**Ritorna**

the call back function or NULL if not set

*lv\_group\_edge\_cb\_t* **lv\_group\_get\_edge\_cb**(const lv\_group\_t \*group)

Get edge callback function of a group.

**Parametri**

**group** -- pointer to a group

**Ritorna**

the call back function or NULL if not set

bool **lv\_group\_get\_editing**(const lv\_group\_t \*group)

Get current mode (edit or navigate).

**Parametri**

**group** -- pointer to group

**Ritorna**

true: edit mode; false: navigate mode

bool **lv\_group\_get\_wrap**(lv\_group\_t \*group)

Get whether moving focus to next/previous Widget will allow wrapping from first->last or last->first Widget.

**Parametri**

**group** -- pointer to group

uint32\_t **lv\_group\_get\_obj\_count**(lv\_group\_t \*group)

Get number of Widgets in group.

**Parametri**

**group** -- pointer to a group

**Ritorna**

number of Widgets in the group

lv\_obj\_t \***lv\_group\_get\_obj\_by\_index**(lv\_group\_t \*group, uint32\_t index)

Get nth Widget within group.

**Parametri**

- **group** -- pointer to a group
- **index** -- index of Widget within the group

**Ritorna**

pointer to Widget

uint32\_t **lv\_group\_get\_count**(void)

Get the number of groups.

**Ritorna**

number of groups

lv\_group\_t \***lv\_group\_by\_index**(uint32\_t index)

Get a group by its index.

**Parametri**

**index** -- index of the group

**Ritorna**

pointer to the group

### 14.10.3 lv\_group\_private.h

#### Functions

void **lv\_group\_init**(void)

Init the group module

**ⓘ Remark**

Internal function, do not call directly.

**void lv\_group\_deinit(void)**

Deinit the group module

 **ⓘ Remark**

Internal function, do not call directly.

**struct \_lv\_group\_t**

#include <lv\_group\_private.h> Groups can be used to logically hold objects so that they can be individually focused. They are NOT for laying out objects on a screen (try layouts for that).

**Public Members**

**lv\_ll\_t obj\_ll**

Linked list to store the objects in the group

**lv\_obj\_t \*\*obj\_focus**

The object in focus

**lv\_group\_focus\_cb\_t focus\_cb**

A function to call when a new object is focused (optional)

**lv\_group\_edge\_cb\_t edge\_cb**

A function to call when an edge is reached, no more focus targets are available in this direction (to allow edge feedback like a sound or a scroll bounce)

**void \*user\_data**

**uint8\_t frozen**

1: can't focus to new object

**uint8\_t editing**

1: Edit mode, 0: Navigate mode

**uint8\_t refocus\_policy**

1: Focus prev if focused on deletion. 0: Focus next if focused on deletion.

**uint8\_t wrap**

1: Focus next/prev can wrap at end of list. 0: Focus next/prev stops at end of list.

## 14.10.4 lv\_obj.h

### Defines

**LV\_ASSERT\_OBJ(obj\_p, obj\_class)**

**LV\_TRACE\_OBJ\_CREATE(...)**

**Enums****enum `lv_obj_flag_t`**

On/Off features controlling the object's behavior. OR-ed values are possible

Note: update obj flags corresponding properties below whenever add/remove flags or change bit definition of flags.

*Values:*

**enumerator `LV_OBJ_FLAG_HIDDEN`**

Make the object hidden. (Like it wasn't there at all)

**enumerator `LV_OBJ_FLAG_CLICKABLE`**

Make the object clickable by the input devices

**enumerator `LV_OBJ_FLAG_CLICK_FOCUSABLE`**

Add focused state to the object when clicked

**enumerator `LV_OBJ_FLAG_CHECKABLE`**

Toggle checked state when the object is clicked

**enumerator `LV_OBJ_FLAG_SCROLLABLE`**

Make the object scrollable

**enumerator `LV_OBJ_FLAG_SCROLL_ELASTIC`**

Allow scrolling inside but with slower speed

**enumerator `LV_OBJ_FLAG_SCROLL_MOMENTUM`**

Make the object scroll further when "thrown"

**enumerator `LV_OBJ_FLAG_SCROLL_ONE`**

Allow scrolling only one snappable children

**enumerator `LV_OBJ_FLAG_SCROLL_CHAIN_HOR`**

Allow propagating the horizontal scroll to a parent

**enumerator `LV_OBJ_FLAG_SCROLL_CHAIN_VER`**

Allow propagating the vertical scroll to a parent

**enumerator `LV_OBJ_FLAG_SCROLL_CHAIN`****enumerator `LV_OBJ_FLAG_SCROLL_ON_FOCUS`**

Automatically scroll object to make it visible when focused

**enumerator `LV_OBJ_FLAG_SCROLL_WITH_ARROW`**

Allow scrolling the focused object with arrow keys

**enumerator `LV_OBJ_FLAG_SNAPPABLE`**

If scroll snap is enabled on the parent it can snap to this object

**enumerator LV\_OBJ\_FLAG\_PRESS\_LOCK**

Keep the object pressed even if the press slid from the object

**enumerator LV\_OBJ\_FLAG\_EVENT\_BUBBLE**

Propagate the events to the parent too

**enumerator LV\_OBJ\_FLAG\_GESTURE\_BUBBLE**

Propagate the gestures to the parent

**enumerator LV\_OBJ\_FLAG\_ADV\_HITTEST**

Allow performing more accurate hit (click) test. E.g. consider rounded corners.

**enumerator LV\_OBJ\_FLAG\_IGNORE\_LAYOUT**

Make the object not positioned by the layouts

**enumerator LV\_OBJ\_FLAG\_FLOATING**

Do not scroll the object when the parent scrolls and ignore layout

**enumerator LV\_OBJ\_FLAG\_SEND\_DRAW\_TASK\_EVENTS**

Send LV\_EVENT\_DRAW\_TASK\_ADDED events

**enumerator LV\_OBJ\_FLAG\_OVERFLOW\_VISIBLE**

Do not clip the children to the parent's ext draw size

**enumerator LV\_OBJ\_FLAG\_EVENT\_TRICKLE**

Propagate the events to the children too

**enumerator LV\_OBJ\_FLAG\_STATE\_TRICKLE**

Propagate the states to the children too

**enumerator LV\_OBJ\_FLAG\_LAYOUT\_1**

Custom flag, free to use by layouts

**enumerator LV\_OBJ\_FLAG\_LAYOUT\_2**

Custom flag, free to use by layouts

**enumerator LV\_OBJ\_FLAG\_FLEX\_IN\_NEW\_TRACK**

Start a new flex track on this item

**enumerator LV\_OBJ\_FLAG\_WIDGET\_1**

Custom flag, free to use by widget

**enumerator LV\_OBJ\_FLAG\_WIDGET\_2**

Custom flag, free to use by widget

**enumerator LV\_OBJ\_FLAG\_USER\_1**

Custom flag, free to use by user

enumerator **LV\_OBJ\_FLAG\_USER\_2**

Custom flag, free to use by user

enumerator **LV\_OBJ\_FLAG\_USER\_3**

Custom flag, free to use by user

enumerator **LV\_OBJ\_FLAG\_USER\_4**

Custom flag, free to use by user

enum **\_lv\_signed\_prop\_id\_t**

*Values:*

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_START**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_HIDDEN**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_CLICKABLE**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_CLICK\_FOCUSABLE**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_CHECKABLE**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_SCROLLABLE**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_SCROLL\_ELASTIC**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_SCROLL\_MOMENTUM**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_SCROLL\_ONE**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_SCROLL\_CHAIN\_HOR**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_SCROLL\_CHAIN\_VER**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_SCROLL\_ON\_FOCUS**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_SCROLL\_WITH\_ARROW**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_SNAPPABLE**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_PRESS\_LOCK**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_EVENT\_BUBBLE**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_GESTURE\_BUBBLE**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_ADV\_HITTEST**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_IGNORE\_LAYOUT**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_FLOATING**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_SEND\_DRAW\_TASK\_EVENTS**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_OVERFLOW\_VISIBLE**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_EVENT\_TRICKLE**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_STATE\_TRICKLE**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_LAYOUT\_1**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_LAYOUT\_2**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_FLEX\_IN\_NEW\_TRACK**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_WIDGET\_1**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_WIDGET\_2**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_USER\_1**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_USER\_2**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_USER\_3**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_USER\_4**

enumerator **LV\_PROPERTY\_OBJ\_FLAG\_END**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_START**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_CHECKED**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_FOCUSED**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_FOCUS\_KEY**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_EDITED**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_HOVERED**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_PRESSED**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_SCROLLLED**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_DISABLED**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_USER\_1**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_USER\_2**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_USER\_3**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_USER\_4**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_ANY**

enumerator **LV\_PROPERTY\_OBJ\_STATE\_END**

enumerator **LV\_PROPERTY\_OBJ\_PARENT**

enumerator **LV\_PROPERTY\_OBJ\_X**

enumerator **LV\_PROPERTY\_OBJ\_Y**

enumerator **LV\_PROPERTY\_OBJ\_W**

enumerator **LV\_PROPERTY\_OBJ\_H**

enumerator **LV\_PROPERTY\_OBJ\_CONTENT\_WIDTH**

enumerator **LV\_PROPERTY\_OBJ\_CONTENT\_HEIGHT**

enumerator **LV\_PROPERTY\_OBJ\_LAYOUT**

enumerator **LV\_PROPERTY\_OBJ\_ALIGN**

enumerator **LV\_PROPERTY\_OBJ\_SCROLLBAR\_MODE**

enumerator **LV\_PROPERTY\_OBJ\_SCROLL\_DIR**

enumerator **LV\_PROPERTY\_OBJ\_SCROLL\_SNAP\_X**

enumerator **LV\_PROPERTY\_OBJ\_SCROLL\_SNAP\_Y**

enumerator **LV\_PROPERTY\_OBJ\_SCROLL\_X**

enumerator **LV\_PROPERTY\_OBJ\_SCROLL\_Y**

enumerator **LV\_PROPERTY\_OBJ\_SCROLL\_TOP**

enumerator **LV\_PROPERTY\_OBJ\_SCROLL\_BOTTOM**

enumerator **LV\_PROPERTY\_OBJ\_SCROLL\_LEFT**

enumerator **LV\_PROPERTY\_OBJ\_SCROLL\_RIGHT**

enumerator **LV\_PROPERTY\_OBJ\_SCROLL\_END**

enumerator **LV\_PROPERTY\_OBJ\_EXT\_DRAW\_SIZE**

enumerator **LV\_PROPERTY\_OBJ\_EVENT\_COUNT**

enumerator **LV\_PROPERTY\_OBJ\_SCREEN**

enumerator **LV\_PROPERTY\_OBJ\_DISPLAY**

enumerator **LV\_PROPERTY\_OBJ\_CHILD\_COUNT**

enumerator **LV\_PROPERTY\_OBJ\_INDEX**

enumerator **LV\_PROPERTY\_OBJ\_END**

## Functions

`lv_obj_t *lv_obj_create(lv_obj_t *parent)`

Create a base object (a rectangle)

**Parametri**

**parent** -- pointer to a parent object. If NULL then a screen will be created.

**Ritorna**

pointer to the new object

`void lv_obj_add_flag(lv_obj_t *obj, lv_obj_flag_t f)`

Set one or more flags

**Parametri**

- **obj** -- pointer to an object
- **f** -- OR-ed values from `lv_obj_flag_t` to set.

`void lv_obj_remove_flag(lv_obj_t *obj, lv_obj_flag_t f)`

Remove one or more flags

**Parametri**

- **obj** -- pointer to an object
- **f** -- OR-ed values from `lv_obj_flag_t` to clear.

`void lv_obj_set_flag(lv_obj_t *obj, lv_obj_flag_t f, bool v)`

Set add or remove one or more flags.

**Parametri**

- **obj** -- pointer to an object
- **f** -- OR-ed values from `lv_obj_flag_t` to update.

- **v** -- true: add the flags; false: remove the flags

**void lv\_obj\_add\_state(lv\_obj\_t \*obj, lv\_state\_t state)**

Add one or more states to the object. The other state bits will remain unchanged. If specified in the styles, transition animation will be started from the previous state to the current.

**Parametri**

- **obj** -- pointer to an object
- **state** -- the states to add. E.g LV\_STATE\_PRESSED | LV\_STATE\_FOCUSED

**void lv\_obj\_remove\_state(lv\_obj\_t \*obj, lv\_state\_t state)**

Remove one or more states to the object. The other state bits will remain unchanged. If specified in the styles, transition animation will be started from the previous state to the current.

**Parametri**

- **obj** -- pointer to an object
- **state** -- the states to add. E.g LV\_STATE\_PRESSED | LV\_STATE\_FOCUSED

**void lv\_obj\_set\_state(lv\_obj\_t \*obj, lv\_state\_t state, bool v)**

Add or remove one or more states to the object. The other state bits will remain unchanged.

**Parametri**

- **obj** -- pointer to an object
- **state** -- the states to add. E.g LV\_STATE\_PRESSED | LV\_STATE\_FOCUSED
- **v** -- true: add the states; false: remove the states

**void lv\_obj\_set\_user\_data(lv\_obj\_t \*obj, void \*user\_data)**

Set the user\_data field of the object

**Parametri**

- **obj** -- pointer to an object
- **user\_data** -- pointer to the new user\_data.

**bool lv\_obj\_has\_flag(const lv\_obj\_t \*obj, lv\_obj\_flag\_t f)**

Check if a given flag or all the given flags are set on an object.

**Parametri**

- **obj** -- pointer to an object
- **f** -- the flag(s) to check (OR-ed values can be used)

**Ritorna**

true: all flags are set; false: not all flags are set

**bool lv\_obj\_has\_flag\_any(const lv\_obj\_t \*obj, lv\_obj\_flag\_t f)**

Check if a given flag or any of the flags are set on an object.

**Parametri**

- **obj** -- pointer to an object
- **f** -- the flag(s) to check (OR-ed values can be used)

**Ritorna**

true: at least one flag is set; false: none of the flags are set

**lv\_state\_t lv\_obj\_get\_state(const lv\_obj\_t \*obj)**

Get the state of an object

**Parametri**

- **obj** -- pointer to an object

**Ritorna**

the state (OR-ed values from *lv\_state\_t*)

**bool lv\_obj\_has\_state(const lv\_obj\_t \*obj, lv\_state\_t state)**

Check if the object is in a given state or not.

**Parametri**

- **obj** -- pointer to an object
- **state** -- a state or combination of states to check

**Ritorna**  
 true: **obj** is in **state**; false: **obj** is not in **state**

**lv\_group\_t \*lv\_obj\_get\_group**(const lv\_obj\_t \*obj)

Get the group of the object

**Parametri**  
**obj** -- pointer to an object

**Ritorna**  
 the pointer to group of the object

**void \*lv\_obj\_get\_user\_data**(lv\_obj\_t \*obj)

Get the user\_data field of the object

**Parametri**  
**obj** -- pointer to an object

**Ritorna**  
 the pointer to the user\_data of the object

**void lv\_obj\_allocate\_spec\_attr**(lv\_obj\_t \*obj)

Allocate special data for an object if not allocated yet.

**Parametri**  
**obj** -- pointer to an object

**bool lv\_obj\_check\_type**(const lv\_obj\_t \*obj, const lv\_obj\_class\_t \*class\_p)

Check the type of obj.

**Parametri**  

- **obj** -- pointer to an object
- **class\_p** -- a class to check (e.g. `lv_slider_class`)

**Ritorna**  
 true: **class\_p** is the **obj** class.

**bool lv\_obj\_has\_class**(const lv\_obj\_t \*obj, const lv\_obj\_class\_t \*class\_p)

Check if any object has a given class (type). It checks the ancestor classes too.

**Parametri**  

- **obj** -- pointer to an object
- **class\_p** -- a class to check (e.g. `lv_slider_class`)

**Ritorna**  
 true: **obj** has the given class

**const lv\_obj\_class\_t \*lv\_obj\_get\_class**(const lv\_obj\_t \*obj)

Get the class (type) of the object

**Parametri**  
**obj** -- pointer to an object

**Ritorna**  
 the class (type) of the object

**bool lv\_obj\_is\_valid**(const lv\_obj\_t \*obj)

Check if any object is still "alive".

**Parametri**  
**obj** -- pointer to an object

**Ritorna**  
 true: valid

**void lv\_obj\_null\_on\_delete**(lv\_obj\_t \*\*obj\_ptr)

Utility to set an object reference to NULL when it gets deleted. The reference should be in a location that will not become invalid during the object's lifetime, i.e. static or allocated.

**Parametri**  
**obj\_ptr** -- a pointer to a pointer to an object

**void lv\_obj\_add\_screen\_load\_event**(lv\_obj\_t \*obj, *lv\_event\_code\_t* trigger, lv\_obj\_t \*screen, *lv\_screen\_load\_anim\_t* anim\_type, uint32\_t duration, uint32\_t delay)

Add an event handler to a widget that will load a screen on a trigger.

**Parametri**

- **obj** -- pointer to widget which should load the screen
- **trigger** -- an event code, e.g. LV\_EVENT\_CLICKED
- **screen** -- the screen to load (must be a valid widget)
- **anim\_type** -- element of `lv_screen_load_anim_t` the screen load animation
- **duration** -- duration of the animation in milliseconds
- **delay** -- delay before the screen load in milliseconds

```
void lv_obj_add_screen_create_event(lv_obj_t *obj, lv_event_code_t trigger, lv_screen_create_cb_t
                                    screen_create_cb, lv_screen_load_anim_t anim_type, uint32_t
                                    duration, uint32_t delay)
```

Add an event handler to a widget that will create a screen on a trigger. The created screen will be deleted when it's unloaded

**Parametri**

- **obj** -- pointer to widget which should load the screen
- **trigger** -- an event code, e.g. LV\_EVENT\_CLICKED
- **screen\_create\_cb** -- a callback to create the screen, e.g. `lv_obj_t * myscreen_create(void)`
- **anim\_type** -- element of `lv_screen_load_anim_t` the screen load animation
- **duration** -- duration of the animation in milliseconds
- **delay** -- delay before the screen load in milliseconds

```
void lv_obj_add_play_timeline_event(lv_obj_t *obj, lv_event_code_t trigger, lv_anim_timeline_t *at,
                                    uint32_t delay, bool reverse)
```

Play a timeline animation on a trigger

**Parametri**

- **obj** -- pointer to widget which should trigger playing the animation
- **trigger** -- an event code, e.g. LV\_EVENT\_CLICKED
- **at** -- pointer to an animation timeline
- **delay** -- wait time before starting the animation
- **reverse** -- true: play in reverse

```
void lv_obj_set_id(lv_obj_t *obj, void *id)
```

Set an id for an object.

**Parametri**

- **obj** -- pointer to an object
- **id** -- the id of the object

```
void *lv_obj_get_id(const lv_obj_t *obj)
```

Get the id of an object.

**Parametri**

**obj** -- pointer to an object

**Ritorna**

the id of the object

```
lv_obj_t *lv_obj_find_by_id(const lv_obj_t *obj, const void *id)
```

DEPRECATED IDs are used only to print the widget trees. To find a widget use `lv_obj_find_by_name`

Get the child object by its id. It will check children and grandchildren recursively. Function `lv_obj_id_compare` is used to matched obj id with given id.

**Parametri**

- **obj** -- pointer to an object
- **id** -- the id of the child object

**Ritorna**

pointer to the child object or NULL if not found

```
void lv_obj_assign_id(const lv_obj_class_t *class_p, lv_obj_t *obj)
```

Assign id to object if not previously assigned. This function gets called automatically when LV\_OBJ\_ID\_AUTO\_ASSIGN is enabled.

Set LV\_USE\_OBJ\_ID\_BUILTIN to use the builtin method to generate object ID. Otherwise, these functions including `lv_obj_[set|assign|free|stringify]_id` and `lv_obj_id_compare` should be implemented externally.

#### Parametri

- **class\_p** -- the class this obj belongs to. Note obj->class\_p is the class currently being constructed.
- **obj** -- pointer to an object

```
void lv_obj_free_id(lv_obj_t *obj)
```

Free resources allocated by `lv_obj_assign_id` or `lv_obj_set_id`. This function is also called automatically when object is deleted.

#### Parametri

- **obj** -- pointer to an object

```
int lv_obj_id_compare(const void *id1, const void *id2)
```

Compare two obj id, return 0 if they are equal.

Set LV\_USE\_OBJ\_ID\_BUILTIN to use the builtin method for compare. Otherwise, it must be implemented externally.

#### Parametri

- **id1** -- the first id
- **id2** -- the second id

#### Ritorna

0 if they are equal, non-zero otherwise.

```
const char *lv_obj_stringify_id(lv_obj_t *obj, char *buf, uint32_t len)
```

Format an object's id into a string.

#### Parametri

- **obj** -- pointer to an object
- **buf** -- buffer to write the string into
- **len** -- length of the buffer

```
void lv_objid_builtin_destroy(void)
```

Free resources used by builtin ID generator.

## Variables

```
const lv_obj_class_t lv_obj_class
```

Make the base object's class publicly available.

## 14.10.5 lv\_obj\_class.h

### Typedef

```
typedef void (*lv_obj_class_event_cb_t)(lv_obj_class_t *class_p, lv_event_t *e)
```

### Enums

```
enum lv_obj_class_editable_t
```

*Values:*

enumerator **LV\_OBJ\_CLASS\_EDITABLE\_INHERIT**

Check the base class. Must have 0 value to let zero initialized class inherit

enumerator **LV\_OBJ\_CLASS\_EDITABLE\_TRUE**

enumerator **LV\_OBJ\_CLASS\_EDITABLE\_FALSE**

enum **lv\_obj\_class\_group\_def\_t**

*Values:*

enumerator **LV\_OBJ\_CLASS\_GROUP\_DEF\_INHERIT**

Check the base class. Must have 0 value to let zero initialized class inherit

enumerator **LV\_OBJ\_CLASS\_GROUP\_DEF\_TRUE**

enumerator **LV\_OBJ\_CLASS\_GROUP\_DEF\_FALSE**

enum **lv\_obj\_class\_theme\_inheritable\_t**

*Values:*

enumerator **LV\_OBJ\_CLASS\_THEME\_INHERITABLE\_FALSE**

Do not inherit theme from base class.

enumerator **LV\_OBJ\_CLASS\_THEME\_INHERITABLE\_TRUE**

## Functions

`lv_obj_t *lv_obj_class_create_obj (const lv_obj_class_t *class_p, lv_obj_t *parent)`

Create an object form a class descriptor

### Parametri

- **class\_p** -- pointer to a class
- **parent** -- pointer to an object where the new object should be created

### Ritorna

pointer to the created object

`void lv_obj_class_init_obj (lv_obj_t *obj)`

`bool lv_obj_is_editable (lv_obj_t *obj)`

`bool lv_obj_is_group_def (lv_obj_t *obj)`

## 14.10.6 lv\_obj\_class\_private.h

### Functions

`void lv_obj_destruct (lv_obj_t *obj)`

struct **\_lv\_obj\_class\_t**

`#include <lv_obj_class_private.h>` Describe the common methods of every object. Similar to a C++ class.

### Public Members

`const lv_obj_class_t *base_class`

```

void (*constructor_cb)(const lv_obj_class_t *class_p, lv_obj_t *obj)
    class_p is the final class while obj->class_p is the class currently being [de]constructed.

void (*destructor_cb)(const lv_obj_class_t *class_p, lv_obj_t *obj)

void (*event_cb)(const lv_obj_class_t *class_p, lv_event_t *e)
    class_p is the class in which event is being processed. Widget type specific event function

uint32_t prop_index_start

uint32_t prop_index_end

const lv_property_ops_t *properties

uint32_t properties_count

const lv_property_name_t *property_names

uint32_t names_count

void *user_data

const char *name

int32_t width_def

int32_t height_def

uint32_t editable
    Value from lv_obj_class_editable_t

uint32_t group_def
    Value from lv_obj_class_group_def_t

uint32_t instance_size

uint32_t theme_inheritable
    Value from lv_obj_class_theme_inheritable_t

```

## 14.10.7 lv\_obj\_draw.h

### Enums

#### enum **lv\_layer\_type\_t**

Store the type of layer required to render a widget.

*Values:*

**enumerator LV\_LAYER\_TYPE\_NONE**

No layer is needed.

**enumerator LV\_LAYER\_TYPE\_SIMPLE**

Simple layer means that the layer can be rendered in chunks. For example with `opa_layered = 140` it's possible to render only 10 lines from the layer. When it's ready go the next 10 lines. It avoids large memory allocations for the layer buffer. The buffer size for a chunk can be set by `LV_DRAW_LAYER_SIMPLE_BUF_SIZE` in `lv_conf.h`.

**enumerator LV\_LAYER\_TYPE\_TRANSFORM**

The widget is transformed and cannot be rendered in chunks. It's because - due to the transformations - pixel outside of a given area will also contribute to the final image. In this case there is no limitation on the buffer size. LVGL will allocate as large buffer as needed to render the transformed area.

**Functions**

**void lv\_obj\_init\_draw\_rect\_dsc(lv\_obj\_t \*obj, lv\_part\_t part, lv\_draw\_rect\_dsc\_t \*draw\_dsc)**

Initialize a rectangle draw descriptor from an object's styles in its current state

**Nota**

Only the relevant fields will be set. E.g. if `border width == 0` the other border properties won't be evaluated.

**Parametri**

- **obj** -- pointer to an object
- **part** -- part of the object, e.g. `LV_PART_MAIN`, `LV_PART_SCROLLBAR`, `LV_PART_KNOB`, etc
- **draw\_dsc** -- the descriptor to initialize. If an `..._opa` field is set to `LV_OPA_TRANSP` the related properties won't be initialized. Should be initialized with `lv_draw_rect_dsc_init(draw_dsc)`.

**void lv\_obj\_init\_draw\_label\_dsc(lv\_obj\_t \*obj, lv\_part\_t part, lv\_draw\_label\_dsc\_t \*draw\_dsc)**

Initialize a label draw descriptor from an object's styles in its current state

**Parametri**

- **obj** -- pointer to an object
- **part** -- part of the object, e.g. `LV_PART_MAIN`, `LV_PART_SCROLLBAR`, `LV_PART_KNOB`, etc
- **draw\_dsc** -- the descriptor to initialize. If the `opa` field is set to or the property is equal to `LV_OPA_TRANSP` the rest won't be initialized. Should be initialized with `lv_draw_label_dsc_init(draw_dsc)`.

**void lv\_obj\_init\_draw\_image\_dsc(lv\_obj\_t \*obj, lv\_part\_t part, lv\_draw\_image\_dsc\_t \*draw\_dsc)**

Initialize an image draw descriptor from an object's styles in its current state

**Parametri**

- **obj** -- pointer to an object
- **part** -- part of the object, e.g. `LV_PART_MAIN`, `LV_PART_SCROLLBAR`, `LV_PART_KNOB`, etc
- **draw\_dsc** -- the descriptor to initialize. Should be initialized with `lv_draw_image_dsc_init(draw_dsc)`.

**void lv\_obj\_init\_draw\_line\_dsc(lv\_obj\_t \*obj, lv\_part\_t part, lv\_draw\_line\_dsc\_t \*draw\_dsc)**

Initialize a line draw descriptor from an object's styles in its current state

**Parametri**

- **obj** -- pointer to an object
- **part** -- part of the object, e.g. `LV_PART_MAIN`, `LV_PART_SCROLLBAR`, `LV_PART_KNOB`, etc

- **draw\_dsc** -- the descriptor to initialize. Should be initialized with `lv_draw_line_dsc_init(draw_dsc)`.

**void lv\_obj\_init\_draw\_arc\_dsc(lv\_obj\_t \*obj, lv\_part\_t part, lv\_draw\_arc\_dsc\_t \*draw\_dsc)**

Initialize an arc draw descriptor from an object's styles in its current state

**Parametri**

- **obj** -- pointer to an object
- **part** -- part of the object, e.g. LV\_PART\_MAIN, LV\_PART\_SCROLLBAR, LV\_PART\_KNOB, etc
- **draw\_dsc** -- the descriptor to initialize. Should be initialized with `lv_draw_arc_dsc_init(draw_dsc)`.

**int32\_t lv\_obj\_calculate\_ext\_draw\_size(lv\_obj\_t \*obj, lv\_part\_t part)**

Get the required extra size (around the object's part) to draw shadow, outline, value etc.

**Parametri**

- **obj** -- pointer to an object
- **part** -- part of the object

**Ritorna**

the extra size required around the object

**void lv\_obj\_refresh\_ext\_draw\_size(lv\_obj\_t \*obj)**

Send a 'LV\_EVENT\_REFR\_EXT\_DRAW\_SIZE' Call the ancestor's event handler to the object to refresh the value of the extended draw size. The result will be saved in **obj**.

**Parametri**

- **obj** -- pointer to an object

## 14.10.8 lv\_obj\_draw\_private.h

### Functions

**int32\_t lv\_obj\_get\_ext\_draw\_size(const lv\_obj\_t \*obj)**

Get the extended draw area of an object.

**Parametri**

- **obj** -- pointer to an object

**Ritorna**

the size extended draw area around the real coordinates

**lv\_layer\_type\_t lv\_obj\_get\_layer\_type(const lv\_obj\_t \*obj)**

## 14.10.9 lv\_obj\_event.h

### Enums

**enum lv\_cover\_res\_t**

Cover check results.

*Values:*

enumerator **LV\_COVER\_RES\_COVER**

enumerator **LV\_COVER\_RES\_NOT\_COVER**

enumerator **LV\_COVER\_RES\_MASKED**

## Functions

`lv_result_t lv_obj_send_event(lv_obj_t *obj, lv_event_code_t event_code, void *param)`

Send an event to the object

**Parametri**

- **obj** -- pointer to an object
- **event\_code** -- the type of the event from `lv_event_t`
- **param** -- arbitrary data depending on the widget type and the event. (Usually NULL)

**Ritorna**

`LV_RESULT_OK`: `obj` was not deleted in the event; `LV_RESULT_INVALID`: `obj` was deleted in the event\_code

`lv_result_t lv_obj_event_base(const lv_obj_class_t *class_p, lv_event_t *e)`

Used by the widgets internally to call the ancestor widget types's event handler

**Parametri**

- **class\_p** -- pointer to the class of the widget (NOT the ancestor class)
- **e** -- pointer to the event descriptor

**Ritorna**

`LV_RESULT_OK`: the target object was not deleted in the event; `LV_RESULT_INVALID`: it was deleted in the event\_code

`lv_obj_t *lv_event_get_current_target_obj(lv_event_t *e)`

Get the current target of the event. It's the object which event handler being called. If the event is not bubbled it's the same as "original" target.

**Parametri**

- **e** -- pointer to the event descriptor

**Ritorna**

the target of the event\_code

`lv_obj_t *lv_event_get_target_obj(lv_event_t *e)`

Get the object originally targeted by the event. It's the same even if the event is bubbled.

**Parametri**

- **e** -- pointer to the event descriptor

**Ritorna**

pointer to the original target of the event\_code

`lv_event_dsc_t *lv_obj_add_event_cb(lv_obj_t *obj, lv_event_cb_t event_cb, lv_event_code_t filter, void *user_data)`

Add an event handler function for an object. Used by the user to react on event which happens with the object. An object can have multiple event handler. They will be called in the same order as they were added.

**Parametri**

- **obj** -- pointer to an object
- **filter** -- an event code (e.g. `LV_EVENT_CLICKED`) on which the event should be called. `LV_EVENT_ALL` can be used to receive all the events.
- **event\_cb** -- the new event function
- **user\_data** -- custom data will be available in `event_cb`

**Ritorna**

handler to the event. It can be used in `lv_obj_remove_event_dsc`.

`uint32_t lv_obj_get_event_count(lv_obj_t *obj)`

`lv_event_dsc_t *lv_obj_get_event_dsc(lv_obj_t *obj, uint32_t index)`

`bool lv_obj_remove_event(lv_obj_t *obj, uint32_t index)`

`bool lv_obj_remove_event_dsc(lv_obj_t *obj, lv_event_dsc_t *dsc)`

`uint32_t lv_obj_remove_event_cb(lv_obj_t *obj, lv_event_cb_t event_cb)`

Remove an event\_cb from an object

**Parametri**

- **obj** -- pointer to a obj

- **event\_cb** -- the event\_cb of the event to remove

**Ritorna**

the count of the event removed

```
uint32_t lv_obj_remove_event_cb_with_user_data(lv_obj_t *obj, lv_event_cb_t event_cb, void *user_data)
```

Remove an event\_cb with user\_data

**Parametri**

- **obj** -- pointer to a obj
- **event\_cb** -- the event\_cb of the event to remove
- **user\_data** -- user\_data

**Ritorna**

the count of the event removed

```
lv_indev_t *lv_event_get_indev(lv_event_t *e)
```

Get the input device passed as parameter to indev related events.

**Parametri**

- **e** -- pointer to an event

**Ritorna**

the indev that triggered the event or NULL if called on a not indev related event

```
lv_layer_t *lv_event_get_layer(lv_event_t *e)
```

Get the draw context which should be the first parameter of the draw functions. Namely: LV\_EVENT\_DRAW\_MAIN/POST, LV\_EVENT\_DRAW\_MAIN/POST\_BEGIN, LV\_EVENT\_DRAW\_MAIN/POST\_END

**Parametri**

- **e** -- pointer to an event

**Ritorna**

pointer to a draw context or NULL if called on an unrelated event

```
const lv_area_t *lv_event_get_old_size(lv_event_t *e)
```

Get the old area of the object before its size was changed. Can be used in LV\_EVENT\_SIZE\_CHANGED

**Parametri**

- **e** -- pointer to an event

**Ritorna**

the old absolute area of the object or NULL if called on an unrelated event

```
uint32_t lv_event_get_key(lv_event_t *e)
```

Get the key passed as parameter to an event. Can be used in LV\_EVENT\_KEY

**Parametri**

- **e** -- pointer to an event

**Ritorna**

the triggering key or NULL if called on an unrelated event

```
int32_t lv_event_get_rotary_diff(lv_event_t *e)
```

Get the signed rotary encoder diff. passed as parameter to an event. Can be used in LV\_EVENT\_ROTARY

**Parametri**

- **e** -- pointer to an event

**Ritorna**

the triggering key or NULL if called on an unrelated event

```
lv_anim_t *lv_event_get_scroll_anim(lv_event_t *e)
```

Get the animation descriptor of a scrolling. Can be used in LV\_EVENT\_SCROLL\_BEGIN

**Parametri**

- **e** -- pointer to an event

**Ritorna**

the animation that will scroll the object. (can be modified as required)

```
void lv_event_set_ext_draw_size(lv_event_t *e, int32_t size)
```

Set the new extra draw size. Can be used in LV\_EVENT\_REFR\_EXT\_DRAW\_SIZE

**Parametri**

- **e** -- pointer to an event
- **size** -- The new extra draw size

`lv_point_t *lv_event_get_self_size_info(lv_event_t *e)`

Get a pointer to an `lv_point_t` variable in which the self size should be saved (width in `point->x` and height `point->y`). Can be used in `LV_EVENT_GET_SELF_SIZE`

**Parametri**

- **e** -- pointer to an event

**Ritorna**

pointer to `lv_point_t` or NULL if called on an unrelated event

`lv_hit_test_info_t *lv_event_get_hit_test_info(lv_event_t *e)`

Get a pointer to an `lv_hit_test_info_t` variable in which the hit test result should be saved. Can be used in `LV_EVENT_HIT_TEST`

**Parametri**

- **e** -- pointer to an event

**Ritorna**

pointer to `lv_hit_test_info_t` or NULL if called on an unrelated event

`const lv_area_t *lv_event_get_cover_area(lv_event_t *e)`

Get a pointer to an area which should be examined whether the object fully covers it or not. Can be used in `LV_EVENT_HIT_TEST`

**Parametri**

- **e** -- pointer to an event

**Ritorna**

an area with absolute coordinates to check

`void lv_event_set_cover_res(lv_event_t *e, lv_cover_res_t res)`

Set the result of cover checking. Can be used in `LV_EVENT_COVER_CHECK`

**Parametri**

- **e** -- pointer to an event
- **res** -- an element of `::lv_cover_check_info_t`

`lv_draw_task_t *lv_event_get_draw_task(lv_event_t *e)`

Get the draw task which was just added. Can be used in `LV_EVENT_DRAW_TASK_ADDED` event

**Parametri**

- **e** -- pointer to an event

**Ritorna**

the added draw task

## 14.10.10 lv\_obj\_event\_private.h

`struct _lv_hit_test_info_t`

`#include <lv_obj_event_private.h>` Used as the event parameter of `LV_EVENT_HIT_TEST` to check if an `point` can click the object or not. `res` should be set like this:

- If already set to `false` another event wants that point non clickable. If you want to respect it leave it as `false` or set `true` to overwrite it.
- If already set `true` and `point` shouldn't be clickable set to `false`
- If already set to `true` you agree that `point` can click the object leave it as `true`

### Public Members

`const lv_point_t *point`

A point relative to screen to check if it can click the object or not

`bool res`

true: point can click the object; false: it cannot

### struct `_lv_cover_check_info_t`

`#include <lv_obj_event_private.h>` Used as the event parameter of `LV_EVENT_COVER_CHECK` to check if an area is covered by the object or not. In the event use `const lv_area_t * area = lv_event_get_cover_area(e)` to get the area to check and `lv_event_set_cover_res(e, res)` to set the result.

#### Public Members

`lv_cover_res_t res`

`const lv_area_t *area`

## 14.10.11 lv\_obj\_pos.h

#### Enums

##### enum `lv_obj_point_transform_flag_t`

Values:

enumerator `LV_OBJ_POINT_TRANSFORM_FLAG_NONE`

No flags

enumerator `LV_OBJ_POINT_TRANSFORM_FLAG_RECURSIVE`

Consider the transformation properties of the parents too

enumerator `LV_OBJ_POINT_TRANSFORM_FLAG_INVERSE`

Execute the inverse of the transformation (-angle and 1/zoom)

enumerator `LV_OBJ_POINT_TRANSFORM_FLAG_INVERSE_RECURSIVE`

Both inverse and recursive

#### Functions

##### void `lv_obj_set_pos(lv_obj_t *obj, int32_t x, int32_t y)`

Set the position of an object relative to the set alignment.

#### Nota

With default alignment it's the distance from the top left corner

#### Nota

E.g. `LV_ALIGN_CENTER` alignment it's the offset from the center of the parent

#### Nota

The position is interpreted on the content area of the parent

**i Nota**

The values can be set in pixel or in percentage of parent size with `lv_pct(v)`

**Parametri**

- **obj** -- pointer to an object
- **x** -- new x coordinate
- **y** -- new y coordinate

`void lv_obj_set_x(lv_obj_t *obj, int32_t x)`

Set the x coordinate of an object

**i Nota**

With default alignment it's the distance from the top left corner

**i Nota**

E.g. `LV_ALIGN_CENTER` alignment it's the offset from the center of the parent

**i Nota**

The position is interpreted on the content area of the parent

**i Nota**

The values can be set in pixel or in percentage of parent size with `lv_pct(v)`

**Parametri**

- **obj** -- pointer to an object
- **x** -- new x coordinate

`void lv_obj_set_y(lv_obj_t *obj, int32_t y)`

Set the y coordinate of an object

**i Nota**

With default alignment it's the distance from the top left corner

**i Nota**

E.g. `LV_ALIGN_CENTER` alignment it's the offset from the center of the parent

**i Nota**

The position is interpreted on the content area of the parent

**Nota**

The values can be set in pixel or in percentage of parent size with `lv_pct(v)`

**Parametri**

- **obj** -- pointer to an object
- **y** -- new y coordinate

`void lv_obj_set_size(lv_obj_t *obj, int32_t w, int32_t h)`

Set the size of an object.

**Nota**

possible values are: pixel simple set the size accordingly LV\_SIZE\_CONTENT set the size to involve all children in the given direction `lv_pct(x)` to set size in percentage of the parent's content area size (the size without paddings). x should be in [0..1000)% range

**Parametri**

- **obj** -- pointer to an object
- **w** -- the new width
- **h** -- the new height

`bool lv_obj_refr_size(lv_obj_t *obj)`

Recalculate the size of the object

**Parametri**

- **obj** -- pointer to an object

**Ritorna**

true: the size has been changed

`void lv_obj_set_width(lv_obj_t *obj, int32_t w)`

Set the width of an object

**Nota**

possible values are: pixel simple set the size accordingly LV\_SIZE\_CONTENT set the size to involve all children in the given direction `lv_pct(x)` to set size in percentage of the parent's content area size (the size without paddings). x should be in [0..1000)% range

**Parametri**

- **obj** -- pointer to an object
- **w** -- the new width

`void lv_obj_set_height(lv_obj_t *obj, int32_t h)`

Set the height of an object

**Nota**

possible values are: pixel simple set the size accordingly LV\_SIZE\_CONTENT set the size to involve all children in the given direction `lv_pct(x)` to set size in percentage of the parent's content area size (the size without paddings). x should be in [0..1000)% range

**Parametri**

- **obj** -- pointer to an object
- **h** -- the new height

`void lv_obj_set_content_width(lv_obj_t *obj, int32_t w)`

Set the width reduced by the left and right padding and the border width.

**Parametri**

- **obj** -- pointer to an object
- **w** -- the width without paddings in pixels

**void lv\_obj\_set\_content\_height(lv\_obj\_t \*obj, int32\_t h)**

Set the height reduced by the top and bottom padding and the border width.

**Parametri**

- **obj** -- pointer to an object
- **h** -- the height without paddings in pixels

**void lv\_obj\_set\_layout(lv\_obj\_t \*obj, uint32\_t layout)**

Set a layout for an object

**Parametri**

- **obj** -- pointer to an object
- **layout** -- pointer to a layout descriptor to set

**bool lv\_obj\_is\_layout\_positioned(const lv\_obj\_t \*obj)**

Test whether the and object is positioned by a layout or not

**Parametri**

- **obj** -- pointer to an object to test

**Ritorna**

true: positioned by a layout; false: not positioned by a layout

**void lv\_obj\_mark\_layout\_as\_dirty(lv\_obj\_t \*obj)**

Mark the object for layout update.

**Parametri**

- **obj** -- pointer to an object whose children need to be updated

**void lv\_obj\_update\_layout(const lv\_obj\_t \*obj)**

Update the layout of an object.

**Parametri**

- **obj** -- pointer to an object whose position and size needs to be updated

**void lv\_obj\_set\_align(lv\_obj\_t \*obj, lv\_align\_t align)**

Change the alignment of an object.

**Parametri**

- **obj** -- pointer to an object to align
- **align** -- type of alignment (see '[lv\\_align\\_t](#)' enum) `LV_ALIGN_OUT_...` can't be used.

**void lv\_obj\_align(lv\_obj\_t \*obj, lv\_align\_t align, int32\_t x\_ofs, int32\_t y\_ofs)**

Change the alignment of an object and set new coordinates. Equivalent to: `lv_obj_set_align(obj, align); lv_obj_set_pos(obj, x_ofs, y_ofs);`

**Parametri**

- **obj** -- pointer to an object to align
- **align** -- type of alignment (see '[lv\\_align\\_t](#)' enum) `LV_ALIGN_OUT_...` can't be used.
- **x\_ofs** -- x coordinate offset after alignment
- **y\_ofs** -- y coordinate offset after alignment

**void lv\_obj\_align\_to(lv\_obj\_t \*obj, const lv\_obj\_t \*base, lv\_align\_t align, int32\_t x\_ofs, int32\_t y\_ofs)**

Align an object to another object.

**Nota**

if the position or size of `base` changes `obj` needs to be aligned manually again

**Parametri**

- **obj** -- pointer to an object to align

- **base** -- pointer to another object (if NULL **obj**'s parent is used). '**obj**' will be aligned to it.
- **align** -- type of alignment (see '*lv\_align\_t*' enum)
- **x\_ofs** -- x coordinate offset after alignment
- **y\_ofs** -- y coordinate offset after alignment

**void lv\_obj\_center(lv\_obj\_t \*obj)**

Align an object to the center on its parent.

#### **Nota**

if the parent size changes **obj** needs to be aligned manually again

#### **Parametri**

**obj** -- pointer to an object to align

**void lv\_obj\_set\_transform(lv\_obj\_t \*obj, const lv\_matrix\_t \*matrix)**

Set the transform matrix of an object

#### **Nota**

`LV_DRAW_TRANSFORM_USE_MATRIX` needs to be enabled.

#### **Parametri**

- **obj** -- pointer to an object
- **matrix** -- pointer to a matrix to set

**void lv\_obj\_reset\_transform(lv\_obj\_t \*obj)**

Reset the transform matrix of an object to identity matrix

#### **Nota**

`LV_DRAW_TRANSFORM_USE_MATRIX` needs to be enabled.

#### **Parametri**

**obj** -- pointer to an object

**void lv\_obj\_get\_coords(const lv\_obj\_t \*obj, lv\_area\_t \*coords)**

Copy the coordinates of an object to an area

#### **Parametri**

- **obj** -- pointer to an object
- **coords** -- pointer to an area to store the coordinates

**int32\_t lv\_obj\_get\_x(const lv\_obj\_t \*obj)**

Get the x coordinate of object.

#### **Nota**

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

#### **Nota**

Zero return value means the object is on the left padding of the parent, and not on the left edge.

**i Nota**

Scrolling of the parent doesn't change the returned value.

**i Nota**

The returned value is always the distance from the parent even if `obj` is positioned by a layout.

**Parametri**

**obj** -- pointer to an object

**Ritorna**

distance of `obj` from the left side of its parent plus the parent's left padding

```
int32_t lv_obj_get_x2(const lv_obj_t *obj)
```

Get the x2 coordinate of object.

**i Nota**

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

**i Nota**

Zero return value means the object is on the right padding of the parent, and not on the right edge.

**i Nota**

Scrolling of the parent doesn't change the returned value.

**i Nota**

The returned value is always the distance from the parent even if `obj` is positioned by a layout.

**Parametri**

**obj** -- pointer to an object

**Ritorna**

distance of `obj` from the right side of its parent plus the parent's right padding

```
int32_t lv_obj_get_y(const lv_obj_t *obj)
```

Get the y coordinate of object.

**i Nota**

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

**i Nota**

Zero return value means the object is on the top padding of the parent, and not on the top edge.

**i Nota**

Scrolling of the parent doesn't change the returned value.

**i Nota**

The returned value is always the distance from the parent even if **obj** is positioned by a layout.

**Parametri**

**obj** -- pointer to an object

**Ritorna**

distance of **obj** from the top side of its parent plus the parent's top padding

`int32_t lv_obj_get_y2(const lv_obj_t *obj)`

Get the y2 coordinate of object.

**i Nota**

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

**i Nota**

Zero return value means the object is on the bottom padding of the parent, and not on the bottom edge.

**i Nota**

Scrolling of the parent doesn't change the returned value.

**i Nota**

The returned value is always the distance from the parent even if **obj** is positioned by a layout.

**Parametri**

**obj** -- pointer to an object

**Ritorna**

distance of **obj** from the bottom side of its parent plus the parent's bottom padding

`int32_t lv_obj_get_x_aligned(const lv_obj_t *obj)`

Get the actually set x coordinate of object, i.e. the offset from the set alignment

**Parametri**

**obj** -- pointer to an object

**Ritorna**

the set x coordinate

`int32_t lv_obj_get_y_aligned(const lv_obj_t *obj)`

Get the actually set y coordinate of object, i.e. the offset from the set alignment

**Parametri**

**obj** -- pointer to an object

**Ritorna**

the set y coordinate

`int32_t lv_obj_get_width(const lv_obj_t *obj)`

Get the width of an object

**Nota**

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

**Parametri**

**obj** -- pointer to an object

**Ritorna**

the width in pixels

`int32_t lv_obj_get_height(const lv_obj_t *obj)`

Get the height of an object

**Nota**

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

**Parametri**

**obj** -- pointer to an object

**Ritorna**

the height in pixels

`int32_t lv_obj_get_content_width(const lv_obj_t *obj)`

Get the width reduced by the left and right padding and the border width.

**Nota**

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

**Parametri**

**obj** -- pointer to an object

**Ritorna**

the width which still fits into its parent without causing overflow (making the parent scrollable)

`int32_t lv_obj_get_content_height(const lv_obj_t *obj)`

Get the height reduced by the top and bottom padding and the border width.

**Nota**

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

**Parametri**

**obj** -- pointer to an object

**Ritorna**

the height which still fits into the parent without causing overflow (making the parent scrollable)

`void lv_obj_get_content_coords(const lv_obj_t *obj, lv_area_t *area)`

Get the area reduced by the paddings and the border width.

**Nota**

The position of the object is recalculated only on the next redraw. To force coordinate recalculation call `lv_obj_update_layout(obj)`.

**Parametri**

- **obj** -- pointer to an object
- **area** -- the area which still fits into the parent without causing overflow (making the parent scrollable)

`int32_t lv_obj_get_self_width(const lv_obj_t *obj)`

Get the width occupied by the "parts" of the widget. E.g. the width of all columns of a table.

**Nota**

This size independent from the real size of the widget. It just tells how large the internal ("virtual") content is.

**Parametri**

- **obj** -- pointer to an objects

**Ritorna**

the width of the virtually drawn content

`int32_t lv_obj_get_self_height(const lv_obj_t *obj)`

Get the height occupied by the "parts" of the widget. E.g. the height of all rows of a table.

**Nota**

This size independent from the real size of the widget. It just tells how large the internal ("virtual") content is.

**Parametri**

- **obj** -- pointer to an objects

**Ritorna**

the width of the virtually drawn content

`bool lv_obj_refresh_self_size(lv_obj_t *obj)`

Handle if the size of the internal ("virtual") content of an object has changed.

**Parametri**

- **obj** -- pointer to an object

**Ritorna**

false: nothing happened; true: refresh happened

`void lv_obj_refr_pos(lv_obj_t *obj)`

`void lv_obj_move_to(lv_obj_t *obj, int32_t x, int32_t y)`

`void lv_obj_move_children_by(lv_obj_t *obj, int32_t x_diff, int32_t y_diff, bool ignore_floating)`

`const lv_matrix_t *lv_obj_get_transform(const lv_obj_t *obj)`

Get the transform matrix of an object

**Parametri**

- **obj** -- pointer to an object

**Ritorna**

pointer to the transform matrix or NULL if not set

`void lv_obj_transform_point(const lv_obj_t *obj, lv_point_t *p, lv_obj_point_transform_flag_t flags)`

Transform a point using the angle and zoom style properties of an object

**Parametri**

- **obj** -- pointer to an object whose style properties should be used
- **p** -- a point to transform, the result will be written back here too
- **flags** -- OR-ed valued of :cpp:enum:`lv_obj_point_transform_flag_t`

`void lv_obj_transform_point_array(const lv_obj_t *obj, lv_point_t points[], size_t count, lv_obj_point_transform_flag_t flags)`

Transform an array of points using the angle and zoom style properties of an object

**Parametri**

- **obj** -- pointer to an object whose style properties should be used
- **points** -- the array of points to transform, the result will be written back here too
- **count** -- number of points in the array
- **flags** -- OR-ed valued of :cpp:enum:`lv_obj_point_transform_flag_t`

`void lv_obj_get_transformed_area(const lv_obj_t *obj, lv_area_t *area, lv_obj_point_transform_flag_t flags)`

Transform an area using the angle and zoom style properties of an object

**Parametri**

- **obj** -- pointer to an object whose style properties should be used
- **area** -- an area to transform, the result will be written back here too
- **flags** -- OR-ed valued of :cpp:enum:`lv_obj_point_transform_flag_t`

`void lv_obj_invalidate_area(const lv_obj_t *obj, const lv_area_t *area)`

Mark an area of an object as invalid. The area will be truncated to the object's area and marked for redraw.

**Parametri**

- **obj** -- pointer to an object
- **area** -- the area to redraw

`void lv_obj_invalidate(const lv_obj_t *obj)`

Mark the object as invalid to redrawn its area

**Parametri**

**obj** -- pointer to an object

`bool lv_obj_area_is_visible(const lv_obj_t *obj, lv_area_t *area)`

Tell whether an area of an object is visible (even partially) now or not

**Parametri**

- **obj** -- pointer to an object
- **area** -- the are to check. The visible part of the area will be written back here.

**Ritorna**

true visible; false not visible (hidden, out of parent, on other screen, etc)

`bool lv_obj_is_visible(const lv_obj_t *obj)`

Tell whether an object is visible (even partially) now or not

**Parametri**

**obj** -- pointer to an object

**Ritorna**

true: visible; false not visible (hidden, out of parent, on other screen, etc)

`void lv_obj_set_ext_click_area(lv_obj_t *obj, int32_t size)`

Set the size of an extended clickable area

**Parametri**

- **obj** -- pointer to an object
- **size** -- extended clickable area in all 4 directions [px]

`void lv_obj_get_click_area(const lv_obj_t *obj, lv_area_t *area)`

Get the an area where to object can be clicked. It's the object's normal area plus the extended click area.

**Parametri**

- **obj** -- pointer to an object
- **area** -- store the result area here

`bool lv_obj_hit_test(lv_obj_t *obj, const lv_point_t *point)`

Hit-test an object given a particular point in screen space.

**Parametri**

- **obj** -- object to hit-test
- **point** -- screen-space point (absolute coordinate)

**Ritorna**

true: if the object is considered under the point

`int32_t lv_clamp_width(int32_t width, int32_t min_width, int32_t max_width, int32_t ref_width)`

Clamp a width between min and max width. If the min/max width is in percentage value use the ref\_width

**Parametri**

- **width** -- width to clamp
- **min\_width** -- the minimal width
- **max\_width** -- the maximal width
- **ref\_width** -- the reference width used when min/max width is in percentage

**Ritorna**

the clamped width

`int32_t lv_clamp_height(int32_t height, int32_t min_height, int32_t max_height, int32_t ref_height)`

Clamp a height between min and max height. If the min/max height is in percentage value use the ref\_height

**Parametri**

- **height** -- height to clamp
- **min\_height** -- the minimal height
- **max\_height** -- the maximal height
- **ref\_height** -- the reference height used when min/max height is in percentage

**Ritorna**

the clamped height

## 14.10.12 lv\_obj\_private.h

`struct _lv_obj_spec_attr_t`

`#include <lv_obj_private.h>` Special, rarely used attributes. They are allocated automatically if any elements is set.

### Public Members

`lv_obj_t **children`

Store the pointer of the children in an array.

`lv_group_t *group_p`

`lv_event_list_t event_list`

`const char *name`

Pointer to the name

`lv_point_t scroll`

The current X/Y scroll offset

`int32_t ext_click_pad`

Extra click padding in all direction

`int32_t ext_draw_size`

EXTend the size in every direction for drawing.

uint16\_t **child\_cnt**

Number of children

uint16\_t **scrollbar\_mode**

How to display scrollbars, see [\*lv\\_scrollbar\\_mode\\_t\*](#)

uint16\_t **scroll\_snap\_x**

Where to align the snappable children horizontally, see [\*lv\\_scroll\\_snap\\_t\*](#)

uint16\_t **scroll\_snap\_y**

Where to align the snappable children vertically

uint16\_t **scroll\_dir**

The allowed scroll direction(s), see [\*lv\\_dir\\_t\*](#)

uint16\_t **layer\_type**

Cache the layer type here. Element of [\*lv\\_intermediate\\_layer\\_type\\_t\*](#)

uint16\_t **name\_static**

1: name was not dynamically allocated

struct **\_lv\_obj\_t**

## Public Members

const lv\_obj\_class\_t \***class\_p**

lv\_obj\_t \***parent**

lv\_obj\_spec\_attr\_t \***spec\_attr**

lv\_obj\_style\_t \***styles**

void \***user\_data**

void \***id**

*lv\_area\_t* **coords**

*lv\_obj\_flag\_t* **flags**

uint16\_t **state**

uint16\_t **layout\_inv**

uint16\_t **readjust\_scroll\_after\_layout**

```
uint16_t scr_layout_inv
```

```
uint16_t skip_trans
```

```
uint16_t style_cnt
```

```
uint16_t h_layout
```

```
uint16_t w_layout
```

```
uint16_t is_deleting
```

### 14.10.13 lv\_obj\_scroll.h

#### Enums

enum **lv\_scrollbar\_mode\_t**

Scrollbar modes: shows when should the scrollbars be visible

*Values:*

enumerator **LV\_SCROLLBAR\_MODE\_OFF**

Never show scrollbars

enumerator **LV\_SCROLLBAR\_MODE\_ON**

Always show scrollbars

enumerator **LV\_SCROLLBAR\_MODE\_ACTIVE**

Show scroll bars when Widget is being scrolled

enumerator **LV\_SCROLLBAR\_MODE\_AUTO**

Show scroll bars when the content is large enough to be scrolled

enum **lv\_scroll\_snap\_t**

Scroll span align options. Tells where to align the snappable children when scroll stops.

*Values:*

enumerator **LV\_SCROLL\_SNAP\_NONE**

Do not align, leave where it is

enumerator **LV\_SCROLL\_SNAP\_START**

Align to the left/top

enumerator **LV\_SCROLL\_SNAP\_END**

Align to the right/bottom

enumerator **LV\_SCROLL\_SNAP\_CENTER**

Align to the center

## Functions

**void `lv_obj_set_scrollbar_mode`(`lv_obj_t` \*obj, `lv_scrollbar_mode_t` mode)**

Set how the scrollbars should behave.

**Parametri**

- **obj** -- pointer to Widget
- **mode** -- LV\_SCROLL\_MODE\_ON/OFF/AUTO/ACTIVE

**void `lv_obj_set_scroll_dir`(`lv_obj_t` \*obj, `lv_dir_t` dir)**

Set direction Widget can be scrolled

**Parametri**

- **obj** -- pointer to Widget
- **dir** -- one or more bit-wise OR-ed values of `lv_dir_t` enumeration

**void `lv_obj_set_scroll_snap_x`(`lv_obj_t` \*obj, `lv_scroll_snap_t` align)**

Set where to snap the children when scrolling ends horizontally

**Parametri**

- **obj** -- pointer to Widget
- **align** -- value from `lv_scroll_snap_t` enumeration

**void `lv_obj_set_scroll_snap_y`(`lv_obj_t` \*obj, `lv_scroll_snap_t` align)**

Set where to snap the children when scrolling ends vertically

**Parametri**

- **obj** -- pointer to Widget
- **align** -- value from `lv_scroll_snap_t` enumeration

**`lv_scrollbar_mode_t lv_obj_get_scrollbar_mode`(const `lv_obj_t` \*obj)**

Get the current scroll mode (when to hide the scrollbars)

**Parametri**

**obj** -- pointer to Widget

**Ritorna**

the current scroll mode from `lv_scrollbar_mode_t`

**`lv_dir_t lv_obj_get_scroll_dir`(const `lv_obj_t` \*obj)**

Get directions Widget can be scrolled (set with `lv_obj_set_scroll_dir()`)

**Parametri**

**obj** -- pointer to Widget

**Ritorna**

current scroll direction bit(s)

**`lv_scroll_snap_t lv_obj_get_scroll_snap_x`(const `lv_obj_t` \*obj)**

Get where to snap child Widgets when horizontal scrolling ends.

**Parametri**

**obj** -- pointer to Widget

**Ritorna**

current snap value from `lv_scroll_snap_t`

**`lv_scroll_snap_t lv_obj_get_scroll_snap_y`(const `lv_obj_t` \*obj)**

Get where to snap child Widgets when vertical scrolling ends.

**Parametri**

**obj** -- pointer to Widget

**Ritorna**

current snap value from `lv_scroll_snap_t`

**`int32_t lv_obj_get_scroll_x`(const `lv_obj_t` \*obj)**

Get current X scroll position. Identical to `lv_obj_get_scroll_left()`.

**Parametri**

**obj** -- pointer to scrollable container Widget

**Ritorna**

current scroll position from left edge

- If Widget is not scrolled return 0.
- If scrolled return > 0.
- If scrolled inside (elastic scroll) return < 0.

`int32_t lv_obj_get_scroll_y(const lv_obj_t *obj)`

Get current Y scroll position. Identical to `lv_obj_get_scroll_top()`.

**Parametri**

**obj** -- pointer to scrollable container Widget

**Ritorna**

current scroll position from top edge

- If Widget is not scrolled return 0.
- If scrolled return > 0.
- If scrolled inside (elastic scroll) return < 0.

`int32_t lv_obj_get_scroll_top(const lv_obj_t *obj)`

Number of pixels a scrollable container Widget can be scrolled down before its top edge appears. When `LV_OBJ_FLAG_SCROLL_ELASTIC` flag is set in Widget, this value can go negative while Widget is being dragged below its normal top-edge boundary.

**Parametri**

**obj** -- pointer to scrollable container Widget

**Ritorna**

pixels Widget can be scrolled down before its top edge appears

`int32_t lv_obj_get_scroll_bottom(const lv_obj_t *obj)`

Number of pixels a scrollable container Widget can be scrolled up before its bottom edge appears. When `LV_OBJ_FLAG_SCROLL_ELASTIC` flag is set in Widget, this value can go negative while Widget is being dragged above its normal bottom-edge boundary.

**Parametri**

**obj** -- pointer to scrollable container Widget

**Ritorna**

pixels Widget can be scrolled up before its bottom edge appears

`int32_t lv_obj_get_scroll_left(const lv_obj_t *obj)`

Number of pixels a scrollable container Widget can be scrolled right before its left edge appears. When `LV_OBJ_FLAG_SCROLL_ELASTIC` flag is set in Widget, this value can go negative while Widget is being dragged farther right than its normal left-edge boundary.

**Parametri**

**obj** -- pointer to scrollable container Widget

**Ritorna**

pixels Widget can be scrolled right before its left edge appears

`int32_t lv_obj_get_scroll_right(const lv_obj_t *obj)`

Number of pixels a scrollable container Widget can be scrolled left before its right edge appears. When `LV_OBJ_FLAG_SCROLL_ELASTIC` flag is set in Widget, this value can go negative while Widget is being dragged farther left than its normal right-edge boundary.

**Parametri**

**obj** -- pointer to scrollable container Widget

**Ritorna**

pixels Widget can be scrolled left before its right edge appears

`void lv_obj_get_scroll_end(lv_obj_t *obj, lv_point_t *end)`

Get the X and Y coordinates where the scrolling will end for Widget if a scrolling animation is in progress. If no scrolling animation, give the current X or y scroll position.

**Parametri**

- **obj** -- pointer to scrollable Widget
- **end** -- pointer to `lv_point_t` object in which to store result

`void lv_obj_scroll_by(lv_obj_t *obj, int32_t dx, int32_t dy, lv_anim_enable_t anim_en)`

Scroll by given amount of pixels.

**Nota**

> 0 value means scroll right/bottom (show the more content on the right/bottom)

**Nota**

e.g. dy = -20 means scroll down 20 px

**Parametri**

- **obj** -- pointer to scrollable Widget to scroll
- **dx** -- pixels to scroll horizontally
- **dy** -- pixels to scroll vertically
- **anim\_en** -- LV\_ANIM\_ON: scroll with animation; LV\_ANIM\_OFF: scroll immediately

**void lv\_obj\_scroll\_by\_bounded(lv\_obj\_t \*obj, int32\_t dx, int32\_t dy, lv\_anim\_enable\_t anim\_en)**

Scroll by given amount of pixels. dx and dy will be limited internally to allow scrolling only on the content area.

**Nota**

e.g. dy = -20 means scroll down 20 px

**Parametri**

- **obj** -- pointer to scrollable Widget to scroll
- **dx** -- pixels to scroll horizontally
- **dy** -- pixels to scroll vertically
- **anim\_en** -- LV\_ANIM\_ON: scroll with animation; LV\_ANIM\_OFF: scroll immediately

**void lv\_obj\_scroll\_to(lv\_obj\_t \*obj, int32\_t x, int32\_t y, lv\_anim\_enable\_t anim\_en)**

Scroll to given coordinate on Widget. X and y will be limited internally to allow scrolling only on the content area.

**Parametri**

- **obj** -- pointer to scrollable Widget to scroll
- **x** -- pixels to scroll horizontally
- **y** -- pixels to scroll vertically
- **anim\_en** -- LV\_ANIM\_ON: scroll with animation; LV\_ANIM\_OFF: scroll immediately

**void lv\_obj\_scroll\_to\_x(lv\_obj\_t \*obj, int32\_t x, lv\_anim\_enable\_t anim\_en)**

Scroll to X coordinate on Widget. X will be limited internally to allow scrolling only on the content area.

**Parametri**

- **obj** -- pointer to scrollable Widget to scroll
- **x** -- pixels to scroll horizontally
- **anim\_en** -- LV\_ANIM\_ON: scroll with animation; LV\_ANIM\_OFF: scroll immediately

**void lv\_obj\_scroll\_to\_y(lv\_obj\_t \*obj, int32\_t y, lv\_anim\_enable\_t anim\_en)**

Scroll to Y coordinate on Widget. y will be limited internally to allow scrolling only on the content area.

**Parametri**

- **obj** -- pointer to scrollable Widget to scroll
- **y** -- pixels to scroll vertically
- **anim\_en** -- LV\_ANIM\_ON: scroll with animation; LV\_ANIM\_OFF: scroll immediately

**void lv\_obj\_scroll\_to\_view(lv\_obj\_t \*obj, *lv\_anim\_enable\_t* anim\_en)**

Scroll **obj**'s parent Widget until **obj** becomes visible.

**Parametri**

- **obj** -- pointer to Widget to scroll into view
- **anim\_en** -- LV\_ANIM\_ON: scroll with animation; LV\_ANIM\_OFF: scroll immediately

**void lv\_obj\_scroll\_to\_view\_recursive(lv\_obj\_t \*obj, *lv\_anim\_enable\_t* anim\_en)**

Scroll **obj**'s parent Widgets recursively until **obj** becomes visible. Widget will be scrolled into view even it has nested scrollable parents.

**Parametri**

- **obj** -- pointer to Widget to scroll into view
- **anim\_en** -- LV\_ANIM\_ON: scroll with animation; LV\_ANIM\_OFF: scroll immediately

**bool lv\_obj\_is\_scrolling(const lv\_obj\_t \*obj)**

Tell whether Widget is being scrolled or not at this moment

**Parametri**

**obj** -- pointer to Widget

**Ritorna**

true: **obj** is being scrolled

**void lv\_obj\_stop\_scroll\_anim(const lv\_obj\_t \*obj)**

Stop scrolling the current object

**Parametri**

**obj** -- The object being scrolled

**void lv\_obj\_update\_snap(lv\_obj\_t \*obj, *lv\_anim\_enable\_t* anim\_en)**

Check children of **obj** and scroll **obj** to fulfill scroll\_snap settings.

**Parametri**

- **obj** -- Widget whose children need to be checked and snapped
- **anim\_en** -- LV\_ANIM\_ON/OFF

**void lv\_obj\_get\_scrollbar\_area(lv\_obj\_t \*obj, *lv\_area\_t* \*hor, *lv\_area\_t* \*ver)**

Get the area of the scrollbars

**Parametri**

- **obj** -- pointer to Widget
- **hor** -- pointer to store the area of the horizontal scrollbar
- **ver** -- pointer to store the area of the vertical scrollbar

**void lv\_obj\_scrollbar\_invalidate(lv\_obj\_t \*obj)**

Invalidate the area of the scrollbars

**Parametri**

**obj** -- pointer to Widget

**void lv\_obj\_readjust\_scroll(lv\_obj\_t \*obj, *lv\_anim\_enable\_t* anim\_en)**

Checks if the content is scrolled "in" and adjusts it to a normal position.

**Parametri**

- **obj** -- pointer to Widget
- **anim\_en** -- LV\_ANIM\_ON/OFF

## 14.10.14 lv\_obj\_scroll\_private.h

### Functions

**lv\_result\_t lv\_obj\_scroll\_by\_raw(lv\_obj\_t \*obj, int32\_t x, int32\_t y)**

Low level function to scroll by given x and y coordinates. LV\_EVENT\_SCROLL is sent.

**Parametri**

- **obj** -- pointer to an object to scroll
- **x** -- pixels to scroll horizontally

- **y** -- pixels to scroll vertically

**Ritorna**

LV\_RESULT\_INVALID: the object was deleted in LV\_EVENT\_SCROLL; LV\_RESULT\_OK: if the object is still valid

## 14.10.15 lv\_obj\_style.h

### Typedef

typedef uint32\_t **lv\_style\_selector\_t**

A joint type for *lv\_part\_t* and *lv\_state\_t*. Example values

- 0: means LV\_PART\_MAIN | LV\_STATE\_DEFAULT
- LV\_STATE\_PRSSED
- LV\_PART\_KNOB
- LV\_PART\_KNOB | LV\_STATE\_PRESSED | LV\_STATE\_CHECKED

### Enums

enum **lv\_state\_t**

Possible states of a widget. OR-ed values are possible

*Values:*

enumerator **LV\_STATE\_DEFAULT**

enumerator **LV\_STATE\_CHECKED**

enumerator **LV\_STATE\_FOCUSED**

enumerator **LV\_STATE\_FOCUS\_KEY**

enumerator **LV\_STATE\_EDITED**

enumerator **LV\_STATE\_HOVERED**

enumerator **LV\_STATE\_PRESSED**

enumerator **LV\_STATE\_SCROLLED**

enumerator **LV\_STATE\_DISABLED**

enumerator **LV\_STATE\_USER\_1**

enumerator **LV\_STATE\_USER\_2**

enumerator **LV\_STATE\_USER\_3**

enumerator **LV\_STATE\_USER\_4**

**enumerator LV\_STATE\_ANY**

Special value can be used in some functions to target all states

**enum lv\_part\_t**

The possible parts of widgets. The parts can be considered as the internal building block of the widgets. E.g. slider = background + indicator + knob Not all parts are used by every widget

*Values:*

**enumerator LV\_PART\_MAIN**

A background like rectangle

**enumerator LV\_PART\_SCROLLBAR**

The scrollbar(s)

**enumerator LV\_PART\_INDICATOR**

Indicator, e.g. for slider, bar, switch, or the tick box of the checkbox

**enumerator LV\_PART\_KNOB**

Like handle to grab to adjust the value

**enumerator LV\_PART\_SELECTED**

Indicate the currently selected option or section

**enumerator LV\_PART\_ITEMS**

Used if the widget has multiple similar elements (e.g. table cells)

**enumerator LV\_PART\_CURSOR**

Mark a specific place e.g. for text area's cursor or on a chart

**enumerator LV\_PART\_CUSTOM\_FIRST**

Extension point for custom widgets

**enumerator LV\_PART\_ANY**

Special value can be used in some functions to target all parts

**enum lv\_style\_state\_cmp\_t**

*Values:*

**enumerator LV\_STYLE\_STATE\_CMP\_SAME**

The style properties in the 2 states are identical

**enumerator LV\_STYLE\_STATE\_CMP\_DIFF\_REDRAW**

The differences can be shown with a simple redraw

**enumerator LV\_STYLE\_STATE\_CMP\_DIFF\_DRAW\_PAD**

The differences can be shown with a simple redraw

**enumerator LV\_STYLE\_STATE\_CMP\_DIFF\_LAYOUT**

The differences can be shown with a simple redraw

## Functions

**void lv\_obj\_add\_style(lv\_obj\_t \*obj, const lv\_style\_t \*style, lv\_style\_selector\_t selector)**

Add a style to an object.

Examples:

```
lv_obj_add_style(btn, &style_btn, 0); //Default button style
lv_obj_add_style(btn, &btn_red, LV_STATE_PRESSED); //Overwrite only some colors to red when pressed
```

### Parametri

- **obj** -- pointer to an object
- **style** -- pointer to a style to add
- **selector** -- OR-ed value of parts and state to which the style should be added

**bool lv\_obj\_replace\_style(lv\_obj\_t \*obj, const lv\_style\_t \*old\_style, const lv\_style\_t \*new\_style, lv\_style\_selector\_t selector)**

Replaces a style of an object, preserving the order of the style stack (local styles and transitions are ignored).

Examples:

```
lv_obj_replace_style(obj, &yellow_style, &blue_style, LV_PART_ANY | LV_STATE_ANY); //Replace a specific style
lv_obj_replace_style(obj, &yellow_style, &blue_style, LV_PART_MAIN | LV_STATE_PRESSED); //Replace a specific style assigned to the main part ↴ when it is pressed
```

### Parametri

- **obj** -- pointer to an object
- **old\_style** -- pointer to a style to replace.
- **new\_style** -- pointer to a style to replace the old style with.
- **selector** -- OR-ed values of states and a part to replace only styles with matching selectors. LV\_STATE\_ANY and LV\_PART\_ANY can be used

**void lv\_obj\_remove\_style(lv\_obj\_t \*obj, const lv\_style\_t \*style, lv\_style\_selector\_t selector)**

Remove a style from an object.

Examples:

```
lv_obj_remove_style(obj, &style, LV_PART_ANY | LV_STATE_ANY); //Remove a specific style
lv_obj_remove_style(obj, NULL, LV_PART_MAIN | LV_STATE_ANY); //Remove all styles from the main part
lv_obj_remove_style(obj, NULL, LV_PART_ANY | LV_STATE_ANY); //Remove all styles
```

### Parametri

- **obj** -- pointer to an object
- **style** -- pointer to a style to remove. Can be NULL to check only the selector
- **selector** -- OR-ed values of states and a part to remove only styles with matching selectors. LV\_STATE\_ANY and LV\_PART\_ANY can be used

**void lv\_obj\_remove\_style\_all(lv\_obj\_t \*obj)**

Remove all styles from an object

### Parametri

- **obj** -- pointer to an object

**void lv\_obj\_report\_style\_change(lv\_style\_t \*style)**

Notify all object if a style is modified

### Parametri

- **style** -- pointer to a style. Only the objects with this style will be notified (NULL to notify all objects)

**void lv\_obj\_refresh\_style(lv\_obj\_t \*obj, lv\_part\_t part, lv\_style\_prop\_t prop)**

Notify an object and its children about its style is modified.

### Parametri

- **obj** -- pointer to an object

- **part** -- the part whose style was changed. E.g. LV\_PART\_ANY, LV\_PART\_MAIN
- **prop** -- LV\_STYLE\_PROP\_ANY or an LV\_STYLE\_... property. It is used to optimize what needs to be refreshed. LV\_STYLE\_PROP\_INV to perform only a style cache update

`void lv_obj_style_set_disabled(lv_obj_t *obj, const lv_style_t *style, lv_style_selector_t selector, bool dis)`

Temporary disable a style for a selector. It will look like is the style wasn't added

**Parametri**

- **obj** -- pointer to an object
- **style** -- pointer to a style
- **selector** -- the selector of a style (e.g. LV\_STATE\_PRESSED | LV\_PART\_KNOB)
- **dis** -- true: disable the style, false: enable the style

`bool lv_obj_style_get_disabled(lv_obj_t *obj, const lv_style_t *style, lv_style_selector_t selector)`

Get if a given style is disabled on an object.

**Parametri**

- **obj** -- pointer to an object
- **style** -- pointer to a style
- **selector** -- the selector of a style (e.g. LV\_STATE\_PRESSED | LV\_PART\_KNOB)

**Ritorna**

true: disable the style, false: enable the style

`void lv_obj_enable_style_refresh(bool en)`

Enable or disable automatic style refreshing when a new style is added/removed to/from an object or any other style change happens.

**Parametri**

- **en** -- true: enable refreshing; false: disable refreshing

`lv_style_value_t lv_obj_get_style_prop(const lv_obj_t *obj, lv_part_t part, lv_style_prop_t prop)`

Get the value of a style property. The current state of the object will be considered. Inherited properties will be inherited. If a property is not set a default value will be returned.

**Parametri**

- **obj** -- pointer to an object
- **part** -- a part from which the property should be get
- **prop** -- the property to get

**Ritorna**

the value of the property. Should be read from the correct field of the `lv_style_value_t` according to the type of the property.

`bool lv_obj_has_style_prop(const lv_obj_t *obj, lv_style_selector_t selector, lv_style_prop_t prop)`

Check if an object has a specified style property for a given style selector.

**Parametri**

- **obj** -- pointer to an object
- **selector** -- the style selector to be checked, defining the scope of the style to be examined.
- **prop** -- the property to be checked.

**Ritorna**

true if the object has the specified selector and property, false otherwise.

`void lv_obj_set_local_style_prop(lv_obj_t *obj, lv_style_prop_t prop, lv_style_value_t value, lv_style_selector_t selector)`

Set local style property on an object's part and state.

**Parametri**

- **obj** -- pointer to an object
- **prop** -- the property
- **value** -- value of the property. The correct element should be set according to the type of the property

- **selector** -- OR-ed value of parts and state for which the style should be set

`lv_style_res_t lv_obj_get_local_style_prop(lv_obj_t *obj, lv_style_prop_t prop, lv_style_value_t *value, lv_style_selector_t selector)`

`bool lv_obj_remove_local_style_prop(lv_obj_t *obj, lv_style_prop_t prop, lv_style_selector_t selector)`

Remove a local style property from a part of an object with a given state.

#### Parametri

- **obj** -- pointer to an object
- **prop** -- a style property to remove.
- **selector** -- OR-ed value of parts and state for which the style should be removed

#### Ritorna

true the property was found and removed; false: the property was not found

`lv_style_value_t lv_obj_style_apply_color_filter(const lv_obj_t *obj, lv_part_t part, lv_style_value_t v)`

Used internally for color filtering

`void lv_obj_fade_in(lv_obj_t *obj, uint32_t time, uint32_t delay)`

Fade in an an object and all its children.

#### Parametri

- **obj** -- the object to fade in
- **time** -- time of fade
- **delay** -- delay to start the animation

`void lv_obj_fade_out(lv_obj_t *obj, uint32_t time, uint32_t delay)`

Fade out an an object and all its children.

#### Parametri

- **obj** -- the object to fade out
- **time** -- time of fade
- **delay** -- delay to start the animation

`static inline lv_state_t lv_obj_style_get_selector_state(lv_style_selector_t selector)`

`static inline lv_part_t lv_obj_style_get_selector_part(lv_style_selector_t selector)`

`static inline void lv_obj_set_style_pad_all(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)`

`static inline void lv_obj_set_style_pad_hor(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)`

`static inline void lv_obj_set_style_pad_ver(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)`

`static inline void lv_obj_set_style_margin_all(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)`

`static inline void lv_obj_set_style_margin_hor(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)`

`static inline void lv_obj_set_style_margin_ver(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)`

`static inline void lv_obj_set_style_pad_gap(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)`

`static inline void lv_obj_set_style_size(lv_obj_t *obj, int32_t width, int32_t height, lv_style_selector_t selector)`

`static inline void lv_obj_set_style_transform_scale(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)`

`static inline int32_t lv_obj_get_style_space_left(const lv_obj_t *obj, lv_part_t part)`

```
static inline int32_t lv_obj_get_style_space_right(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_space_top(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_space_bottom(const lv_obj_t *obj, lv_part_t part)
lv_text_align_t lv_obj_calculate_style_text_align(const lv_obj_t *obj, lv_part_t part, const char
*txt)

static inline int32_t lv_obj_get_style_transform_scale_x_safe(const lv_obj_t *obj, lv_part_t
part)
static inline int32_t lv_obj_get_style_transform_scale_y_safe(const lv_obj_t *obj, lv_part_t
part)
```

**lv\_opa\_t lv\_obj\_get\_style\_opa\_recursive**(const lv\_obj\_t \*obj, *lv\_part\_t* part)

Get the **opa** style property from all parents and multiply and >> 8 them.

**Parametri**

- **obj** -- the object whose opacity should be get
- **part** -- the part whose opacity should be get. Non-MAIN parts will consider the **opa** of the MAIN part too

**Ritorna**

the final opacity considering the parents' opacity too

*lv\_color32\_t* **lv\_obj\_style\_apply\_recolor**(const lv\_obj\_t \*obj, *lv\_part\_t* part, *lv\_color32\_t* color)

Apply recolor effect to the input color based on the object's style properties.

**Parametri**

- **obj** -- the target object containing recolor style properties
- **part** -- the part to retrieve recolor styles.
- **color** -- the original color to be modified

**Ritorna**

the blended color after applying recolor and opacity

*lv\_color32\_t* **lv\_obj\_get\_style\_recolor\_recursive**(const lv\_obj\_t \*obj, *lv\_part\_t* part)

Get the **recolor** style property from all parents and blend them recursively.

**Parametri**

- **obj** -- the object whose recolor value should be retrieved
- **part** -- the target part to check. Non-MAIN parts will also consider the **recolor** value from the MAIN part during calculation

**Ritorna**

the final blended recolor value combining all parent's recolor values

## 14.10.16 lv\_obj\_style\_gen.h

### Functions

```
static inline int32_t lv_obj_get_style_width(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_min_width(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_max_width(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_height(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_min_height(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_max_height(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_length(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_x(const lv_obj_t *obj, lv_part_t part)
```

```

static inline int32_t lv_obj_get_style_y(const lv_obj_t *obj, lv_part_t part)
static inline lv_align_t lv_obj_get_style_align(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_width(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_height(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_translate_x(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_translate_y(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_translate_radial(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_scale_x(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_scale_y(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_rotation(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_pivot_x(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_pivot_y(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_skew_x(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_transform_skew_y(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_top(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_bottom(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_left(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_right(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_row(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_column(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_pad_radial(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_margin_top(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_margin_bottom(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_margin_left(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_margin_right(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_bg_color(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_bg_color_filtered(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_bg_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_bg_grad_color(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_bg_grad_color_filtered(const lv_obj_t *obj, lv_part_t part)
static inline lv_grad_dir_t lv_obj_get_style_bg_grad_dir(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_bg_main_stop(const lv_obj_t *obj, lv_part_t part)

```

```

static inline int32_t lv_obj_get_style_bg_grad_stop(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_bg_main_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_bg_grad_opa(const lv_obj_t *obj, lv_part_t part)
static inline const lv_grad_dsc_t *lv_obj_get_style_bg_grad(const lv_obj_t *obj, lv_part_t part)
static inline const void *lv_obj_get_style_bg_image_src(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_bg_image_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_bg_image_recolor(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_bg_image_recolor_filtered(const lv_obj_t *obj,
                                                               lv_part_t part)

static inline lv_opa_t lv_obj_get_style_bg_image_recolor_opa(const lv_obj_t *obj, lv_part_t part)
static inline bool lv_obj_get_style_bg_image_tiled(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_border_color(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_border_color_filtered(const lv_obj_t *obj, lv_part_t
                                                               part)

static inline lv_opa_t lv_obj_get_style_border_opa(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_border_width(const lv_obj_t *obj, lv_part_t part)
static inline lv_border_side_t lv_obj_get_style_border_side(const lv_obj_t *obj, lv_part_t part)
static inline bool lv_obj_get_style_border_post(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_outline_width(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_outline_color(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_outline_color_filtered(const lv_obj_t *obj, lv_part_t
                                                               part)

static inline lv_opa_t lv_obj_get_style_outline_opa(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_outline_pad(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_shadow_width(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_shadow_offset_x(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_shadow_offset_y(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_shadow_spread(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_shadow_color(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_shadow_color_filtered(const lv_obj_t *obj, lv_part_t
                                                               part)

static inline lv_opa_t lv_obj_get_style_shadow_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_image_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_image_recolor(const lv_obj_t *obj, lv_part_t part)

```

```

static inline lv_color_t lv_obj_get_style_image_recolor_filtered(const lv_obj_t *obj, lv_part_t
part)

static inline lv_opa_t lv_obj_get_style_image_recolor_opa(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_line_width(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_line_dash_width(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_line_dash_gap(const lv_obj_t *obj, lv_part_t part)

static inline bool lv_obj_get_style_line_rounded(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_line_color(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_line_color_filtered(const lv_obj_t *obj, lv_part_t part)

static inline lv_opa_t lv_obj_get_style_line_opa(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_arc_width(const lv_obj_t *obj, lv_part_t part)

static inline bool lv_obj_get_style_arc_rounded(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_arc_color(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_arc_color_filtered(const lv_obj_t *obj, lv_part_t part)

static inline lv_opa_t lv_obj_get_style_arc_opa(const lv_obj_t *obj, lv_part_t part)

static inline const void *lv_obj_get_style_arc_image_src(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_text_color(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_text_color_filtered(const lv_obj_t *obj, lv_part_t part)

static inline lv_opa_t lv_obj_get_style_text_opa(const lv_obj_t *obj, lv_part_t part)

static inline const lv_font_t *lv_obj_get_style_text_font(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_text_letter_space(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_text_line_space(const lv_obj_t *obj, lv_part_t part)

static inline lv_text_decor_t lv_obj_get_style_text_decor(const lv_obj_t *obj, lv_part_t part)

static inline lv_text_align_t lv_obj_get_style_text_align(const lv_obj_t *obj, lv_part_t part)

static inline lv_color_t lv_obj_get_style_text_outline_stroke_color(const lv_obj_t *obj,
lv_part_t part)

static inline lv_color_t lv_obj_get_style_text_outline_stroke_color_filtered(const
lv_obj_t
*obj,
lv_part_t
part)

static inline int32_t lv_obj_get_style_text_outline_stroke_width(const lv_obj_t *obj, lv_part_t
part)

static inline lv_opa_t lv_obj_get_style_text_outline_stroke_opa(const lv_obj_t *obj, lv_part_t
part)

static inline int32_t lv_obj_get_style_radius(const lv_obj_t *obj, lv_part_t part)

```

```

static inline int32_t lv_obj_get_style_radial_offset(const lv_obj_t *obj, lv_part_t part)
static inline bool lv_obj_get_style_clip_corner(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_opa_layered(const lv_obj_t *obj, lv_part_t part)
static inline const lv_color_filter_dsc_t *lv_obj_get_style_color_filter_dsc(const lv_obj_t *obj,
                                                                           lv_part_t part)
static inline lv_opa_t lv_obj_get_style_color_filter_opa(const lv_obj_t *obj, lv_part_t part)
static inline lv_color_t lv_obj_get_style_recolor(const lv_obj_t *obj, lv_part_t part)
static inline lv_opa_t lv_obj_get_style_recolor_opa(const lv_obj_t *obj, lv_part_t part)
static inline const lv_anim_t *lv_obj_get_style_anim(const lv_obj_t *obj, lv_part_t part)
static inline uint32_t lv_obj_get_style_anim_duration(const lv_obj_t *obj, lv_part_t part)
static inline const lv_style_transition_dsc_t *lv_obj_get_style_transition(const lv_obj_t *obj,
                                                                       lv_part_t part)
static inline lv_blend_mode_t lv_obj_get_style_blend_mode(const lv_obj_t *obj, lv_part_t part)
static inline uint16_t lv_obj_get_style_layout(const lv_obj_t *obj, lv_part_t part)
static inline lv_base_dir_t lv_obj_get_style_base_dir(const lv_obj_t *obj, lv_part_t part)
static inline const void *lv_obj_get_style_bitmap_mask_src(const lv_obj_t *obj, lv_part_t part)
static inline uint32_t lv_obj_get_style_rotary_sensitivity(const lv_obj_t *obj, lv_part_t part)
static inline lv_flex_flow_t lv_obj_get_style_flex_flow(const lv_obj_t *obj, lv_part_t part)
static inline lv_flex_align_t lv_obj_get_style_flex_main_place(const lv_obj_t *obj, lv_part_t part)
static inline lv_flex_align_t lv_obj_get_style_flex_cross_place(const lv_obj_t *obj, lv_part_t part)
static inline lv_flex_align_t lv_obj_get_style_flex_track_place(const lv_obj_t *obj, lv_part_t part)
static inline uint8_t lv_obj_get_style_flex_grow(const lv_obj_t *obj, lv_part_t part)
static inline const int32_t *lv_obj_get_style_grid_column_dsc_array(const lv_obj_t *obj,
                                                                    lv_part_t part)
static inline lv_grid_align_t lv_obj_get_style_grid_column_align(const lv_obj_t *obj, lv_part_t
                                                                part)
static inline const int32_t *lv_obj_get_style_grid_row_dsc_array(const lv_obj_t *obj, lv_part_t
                                                                part)
static inline lv_grid_align_t lv_obj_get_style_grid_row_align(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_grid_cell_column_pos(const lv_obj_t *obj, lv_part_t part)
static inline lv_grid_align_t lv_obj_get_style_grid_cell_x_align(const lv_obj_t *obj, lv_part_t
                                                                part)
static inline int32_t lv_obj_get_style_grid_cell_column_span(const lv_obj_t *obj, lv_part_t part)
static inline int32_t lv_obj_get_style_grid_cell_row_pos(const lv_obj_t *obj, lv_part_t part)

```

```

static inline lv_grid_align_t lv_obj_get_style_grid_cell_y_align(const lv_obj_t *obj, lv_part_t part)

static inline int32_t lv_obj_get_style_grid_cell_row_span(const lv_obj_t *obj, lv_part_t part)

void lv_obj_set_style_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_min_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_max_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_height(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_min_height(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_max_height(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_length(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_align(lv_obj_t *obj, lv_align_t value, lv_style_selector_t selector)

void lv_obj_set_style_transform_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_transform_height(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_translate_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_translate_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_translate_radial(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_transform_scale_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_transform_scale_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_transform_rotation(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_transform_pivot_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_transform_pivot_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_transform_skew_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_transform_skew_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_pad_top(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_pad_bottom(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_pad_left(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_pad_right(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_pad_row(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_pad_column(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_pad_radial(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_margin_top(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

```

```

void lv_obj_set_style_margin_bottom(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_margin_left(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_margin_right(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_grad_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_grad_dir(lv_obj_t *obj, lv_grad_dir_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_main_stop(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_grad_stop(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_main_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_grad_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_grad(lv_obj_t *obj, const lv_grad_dsc_t *value, lv_style_selector_t selector)
void lv_obj_set_style_bg_image_src(lv_obj_t *obj, const void *value, lv_style_selector_t selector)
void lv_obj_set_style_bg_image_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_bg_image_recolor(lv_obj_t *obj, lv_color_t value, lv_style_selector_t
                                      selector)
void lv_obj_set_style_bg_image_recolor_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t
                                          selector)
void lv_obj_set_style_bg_image_tiled(lv_obj_t *obj, bool value, lv_style_selector_t selector)
void lv_obj_set_style_border_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_border_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_border_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_border_side(lv_obj_t *obj, lv_border_side_t value, lv_style_selector_t selector)
void lv_obj_set_style_border_post(lv_obj_t *obj, bool value, lv_style_selector_t selector)
void lv_obj_set_style_outline_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_outline_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_outline_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_outline_pad(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_offset_x(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_offset_y(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_spread(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_shadow_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)

```

```

void lv_obj_set_style_image_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_image_recolor(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_image_recolor_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t
                                      selector)

void lv_obj_set_style_line_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_line_dash_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_line_dash_gap(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_line_rounded(lv_obj_t *obj, bool value, lv_style_selector_t selector)
void lv_obj_set_style_line_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_line_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_arc_width(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_arc_rounded(lv_obj_t *obj, bool value, lv_style_selector_t selector)
void lv_obj_set_style_arc_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_arc_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_arc_image_src(lv_obj_t *obj, const void *value, lv_style_selector_t selector)
void lv_obj_set_style_text_color(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_font(lv_obj_t *obj, const lv_font_t *value, lv_style_selector_t selector)
void lv_obj_set_style_text_letter_space(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_line_space(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_decor(lv_obj_t *obj, lv_text_decor_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_align(lv_obj_t *obj, lv_text_align_t value, lv_style_selector_t selector)
void lv_obj_set_style_text_outline_stroke_color(lv_obj_t *obj, lv_color_t value,
                                                lv_style_selector_t selector)

void lv_obj_set_style_text_outline_stroke_width(lv_obj_t *obj, int32_t value,
                                              lv_style_selector_t selector)

void lv_obj_set_style_text_outline_stroke_opa(lv_obj_t *obj, lv_opa_t value,
                                              lv_style_selector_t selector)

void lv_obj_set_style_radius(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_radial_offset(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)
void lv_obj_set_style_clip_corner(lv_obj_t *obj, bool value, lv_style_selector_t selector)
void lv_obj_set_style_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_opa_layered(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_color_filter_dsc(lv_obj_t *obj, const lv_color_filter_dsc_t *value,
                                       lv_style_selector_t selector)

```

```

void lv_obj_set_style_color_filter_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_recolor(lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
void lv_obj_set_style_recolor_opa(lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)
void lv_obj_set_style_anim(lv_obj_t *obj, const lv_anim_t *value, lv_style_selector_t selector)
void lv_obj_set_style_anim_duration(lv_obj_t *obj, uint32_t value, lv_style_selector_t selector)
void lv_obj_set_style_transition(lv_obj_t *obj, const lv_style_transition_dsc_t *value,
                                lv_style_selector_t selector)

void lv_obj_set_style_blend_mode(lv_obj_t *obj, lv_blend_mode_t value, lv_style_selector_t selector)
void lv_obj_set_style_layout(lv_obj_t *obj, uint16_t value, lv_style_selector_t selector)
void lv_obj_set_style_base_dir(lv_obj_t *obj, lv_base_dir_t value, lv_style_selector_t selector)
void lv_obj_set_style_bitmap_mask_src(lv_obj_t *obj, const void *value, lv_style_selector_t
                                      selector)

void lv_obj_set_style_rotary_sensitivity(lv_obj_t *obj, uint32_t value, lv_style_selector_t
                                         selector)

void lv_obj_set_style_flex_flow(lv_obj_t *obj, lv_flex_flow_t value, lv_style_selector_t selector)
void lv_obj_set_style_flex_main_place(lv_obj_t *obj, lv_flex_align_t value, lv_style_selector_t
                                      selector)

void lv_obj_set_style_flex_cross_place(lv_obj_t *obj, lv_flex_align_t value, lv_style_selector_t
                                       selector)

void lv_obj_set_style_flex_track_place(lv_obj_t *obj, lv_flex_align_t value, lv_style_selector_t
                                       selector)

void lv_obj_set_style_flex_grow(lv_obj_t *obj, uint8_t value, lv_style_selector_t selector)
void lv_obj_set_style_grid_column_dsc_array(lv_obj_t *obj, const int32_t *value,
                                            lv_style_selector_t selector)

void lv_obj_set_style_grid_column_align(lv_obj_t *obj, lv_grid_align_t value, lv_style_selector_t
                                         selector)

void lv_obj_set_style_grid_row_dsc_array(lv_obj_t *obj, const int32_t *value, lv_style_selector_t
                                         selector)

void lv_obj_set_style_grid_row_align(lv_obj_t *obj, lv_grid_align_t value, lv_style_selector_t
                                         selector)

void lv_obj_set_style_grid_cell_column_pos(lv_obj_t *obj, int32_t value, lv_style_selector_t
                                         selector)

void lv_obj_set_style_grid_cell_x_align(lv_obj_t *obj, lv_grid_align_t value, lv_style_selector_t
                                         selector)

void lv_obj_set_style_grid_cell_column_span(lv_obj_t *obj, int32_t value, lv_style_selector_t
                                         selector)

void lv_obj_set_style_grid_cell_row_pos(lv_obj_t *obj, int32_t value, lv_style_selector_t selector)

void lv_obj_set_style_grid_cell_y_align(lv_obj_t *obj, lv_grid_align_t value, lv_style_selector_t
                                         selector)

```

---

```
void lv_obj_set_style_grid_cell_row_span(lv_obj_t *obj, int32_t value, lv_style_selector_t
                                         selector)
```

## 14.10.17 lv\_obj\_style\_private.h

### Functions

```
void lv_obj_style_init(void)
```

Initialize the object related style manager module. Called by LVGL in `lv_init()`

```
void lv_obj_style_deinit(void)
```

Deinitialize the object related style manager module. Called by LVGL in `lv_deinit()`

```
void lv_obj_style_create_transition(lv_obj_t *obj, lv_part_t part, lv_state_t prev_state, lv_state_t
                                    new_state, const lv_obj_style_transition_dsc_t *tr)
```

Used internally to create a style transition

#### Parametri

- **obj** --
- **part** --
- **prev\_state** --
- **new\_state** --
- **tr** --

```
lv_style_state_cmp_t lv_obj_style_state_compare(lv_obj_t *obj, lv_state_t state1, lv_state_t state2)
```

Used internally to compare the appearance of an object in 2 states

#### Parametri

- **obj** --
- **state1** --
- **state2** --

#### Ritorna

```
void lv_obj_update_layer_type(lv_obj_t *obj)
```

Update the layer type of a widget based on its current styles. The result will be stored in `obj->spec_attr->layer_type`

#### Parametri

- obj** -- the object whose layer should be updated

```
struct _lv_obj_style_t
```

### Public Members

```
const lv_style_t *style
```

```
uint32_t selector
```

```
uint32_t is_local
```

```
uint32_t is_trans
```

```
uint32_t is_disabled
```

```
struct _lv_obj_style_transition_dsc_t
```

**Public Members**

```
uint16_t time  

uint16_t delay  

lv_style_selector_t selector  

lv_style_prop_t prop  

lv_anim_path_cb_t path_cb  

void *user_data
```

**14.10.18 lv\_obj\_tree.h****Typedef**

```
typedef lv_obj_tree_walk_res_t (*lv_obj_tree_walk_cb_t)(lv_obj_t*, void*)
```

**Enums**

```
enum lv_obj_tree_walk_res_t
```

*Values:*

enumerator **LV\_OBJ\_TREE\_WALK\_NEXT**

enumerator **LV\_OBJ\_TREE\_WALK\_SKIP\_CHILDREN**

enumerator **LV\_OBJ\_TREE\_WALK\_END**

**Functions**

```
void lv_obj_delete(lv_obj_t *obj)
```

Delete an object and all of its children. Also remove the objects from their group and remove all animations (if any). Send **LV\_EVENT\_DELETED** to deleted objects.

**Parametri**

**obj** -- pointer to an object

```
void lv_obj_clean(lv_obj_t *obj)
```

Delete all children of an object. Also remove the objects from their group and remove all animations (if any). Send **LV\_EVENT\_DELETED** to deleted objects.

**Parametri**

**obj** -- pointer to an object

```
void lv_obj_delete_delayed(lv_obj_t *obj, uint32_t delay_ms)
```

Delete an object after some delay

**Parametri**

- **obj** -- pointer to an object
- **delay\_ms** -- time to wait before delete in milliseconds

**void lv\_obj\_delete\_anim\_completed\_cb(lv\_anim\_t \*a)**

A function to be easily used in animation ready callback to delete an object when the animation is ready

**Parametri**

**a** -- pointer to the animation

**void lv\_obj\_delete\_async(lv\_obj\_t \*obj)**

Helper function for asynchronously deleting objects. Useful for cases where you can't delete an object directly in an LV\_EVENT\_DELETE handler (i.e. parent).

 **Vedi anche**

*lv\_async\_call*

**Parametri**

**obj** -- object to delete

**void lv\_obj\_set\_parent(lv\_obj\_t \*obj, lv\_obj\_t \*parent)**

Move the parent of an object. The relative coordinates will be kept.

**Parametri**

- **obj** -- pointer to an object whose parent needs to be changed
- **parent** -- pointer to the new parent

**void lv\_obj\_swap(lv\_obj\_t \*obj1, lv\_obj\_t \*obj2)**

Swap the positions of two objects. When used in listboxes, it can be used to sort the listbox items.

**Parametri**

- **obj1** -- pointer to the first object
- **obj2** -- pointer to the second object

**void lv\_obj\_move\_to\_index(lv\_obj\_t \*obj, int32\_t index)**

moves the object to the given index in its parent. When used in listboxes, it can be used to sort the listbox items.

 **Nota**

to move to the background: `lv_obj_move_to_index(obj, 0)`

 **Nota**

to move forward (up): `lv_obj_move_to_index(obj, lv_obj_get_index(obj) - 1)`

**Parametri**

- **obj** -- pointer to the object to be moved.
- **index** -- new index in parent. -1 to count from the back

**lv\_obj\_t \*lv\_obj\_get\_screen(const lv\_obj\_t \*obj)**

Get the screen of an object

**Parametri**

**obj** -- pointer to an object

**Ritorna**

pointer to the object's screen

**lv\_display\_t \*lv\_obj\_get\_display(const lv\_obj\_t \*obj)**

Get the display of the object

**Parametri**

**obj** -- pointer to an object

**Ritorna**

pointer to the object's display

`lv_obj_t *lv_obj_get_parent(const lv_obj_t *obj)`

Get the parent of an object

**Parametri**

- **obj** -- pointer to an object

**Ritorna**

the parent of the object. (NULL if **obj** was a screen)

`lv_obj_t *lv_obj_get_child(const lv_obj_t *obj, int32_t idx)`

Get the child of an object by the child's index.

**Parametri**

- **obj** -- pointer to an object whose child should be get
- **idx** -- the index of the child. 0: the oldest (firstly created) child 1: the second oldest child count-1: the youngest -1: the youngest -2: the second youngest

**Ritorna**

pointer to the child or NULL if the index was invalid

`lv_obj_t *lv_obj_get_child_by_type(const lv_obj_t *obj, int32_t idx, const lv_obj_class_t *class_p)`

Get the child of an object by the child's index. Consider the children only with a given type.

**Parametri**

- **obj** -- pointer to an object whose child should be get
- **idx** -- the index of the child. 0: the oldest (firstly created) child 1: the second oldest child count-1: the youngest -1: the youngest -2: the second youngest
- **class\_p** -- the type of the children to check

**Ritorna**

pointer to the child or NULL if the index was invalid

`lv_obj_t *lv_obj_get_sibling(const lv_obj_t *obj, int32_t idx)`

Return a sibling of an object

**Parametri**

- **obj** -- pointer to an object whose sibling should be get
- **idx** -- 0: **obj** itself -1: the first older sibling -2: the next older sibling 1: the first younger sibling 2: the next younger sibling etc

**Ritorna**

pointer to the requested sibling or NULL if there is no such sibling

`lv_obj_t *lv_obj_get_sibling_by_type(const lv_obj_t *obj, int32_t idx, const lv_obj_class_t *class_p)`

Return a sibling of an object. Consider the siblings only with a given type.

**Parametri**

- **obj** -- pointer to an object whose sibling should be get
- **idx** -- 0: **obj** itself -1: the first older sibling -2: the next older sibling 1: the first younger sibling 2: the next younger sibling etc
- **class\_p** -- the type of the children to check

**Ritorna**

pointer to the requested sibling or NULL if there is no such sibling

`uint32_t lv_obj_get_child_count(const lv_obj_t *obj)`

Get the number of children

**Parametri**

- **obj** -- pointer to an object

**Ritorna**

the number of children

`uint32_t lv_obj_get_child_count_by_type(const lv_obj_t *obj, const lv_obj_class_t *class_p)`

Get the number of children having a given type.

**Parametri**

- **obj** -- pointer to an object
- **class\_p** -- the type of the children to check

**Ritorna**

the number of children

`void lv_obj_set_name(lv_obj_t *obj, const char *name)`

Set a name for a widget. The name will be allocated and freed when the widget is deleted or a new name is set.

#### Nota

If the name ends with a #, older siblings with the same name will be counted, and the # will be replaced by the index of the given widget. For example, creating multiple widgets with the name "mybtn\_#" will result in resolved names like "mybtn\_0", "mybtn\_1", "mybtn\_2", etc. The name is resolved when `lv_obj_get_name_resolved` is called, so the result reflects the currently existing widgets at that time.

#### Parametri

- **obj** -- pointer to an object
- **name** -- the name to set. If set to `NULL` the default "<widget\_type>\_#" name will be used.

`void lv_obj_set_name_static(lv_obj_t *obj, const char *name)`

Set a name for a widget. Only a pointer will be saved.

#### Nota

If the name ends with a #, older siblings with the same name will be counted, and the # will be replaced by the index of the given widget. For example, creating multiple widgets with the name "mybtn\_#" will result in resolved names like "mybtn\_0", "mybtn\_1", "mybtn\_2", etc. The name is resolved when `lv_obj_get_name_resolved` is called, so the result reflects the currently existing widgets at that time.

#### Parametri

- **obj** -- pointer to an object
- **name** -- the name to set. If set to `NULL` the default "<widget\_type>\_#" name will be used.

`const char *lv_obj_get_name(const lv_obj_t *obj)`

Get the set name as it was set.

#### Parametri

**obj** -- pointer to an object

#### Ritorna

get the set name or `NULL` if it wasn't set yet

`void lv_obj_get_name_resolved(const lv_obj_t *obj, char buf[], size_t buf_size)`

Get the set name or craft a name automatically.

#### Nota

If the name ends with a #, older siblings with the same name will be counted, and the # will be replaced by the index of the given widget. For example, creating multiple widgets with the name "mybtn\_#" will result in resolved names like "mybtn\_0", "mybtn\_1", "mybtn\_2", etc. The name is resolved when `lv_obj_get_name_resolved` is called, so the result reflects the currently existing widgets at that time.

#### Parametri

- **obj** -- pointer to an object
- **buf** -- buffer to store the name
- **buf\_size** -- the size of the buffer in bytes

`lv_obj_t *lv_obj_find_by_name(const lv_obj_t *parent, const char *name)`

Find a child with a given name on a parent. This child doesn't have to be the direct child of the parent. First

direct children of the parent will be checked, and the direct children of the first child, etc. (Breadth-first search).

If the name of a widget was not set a name like "lv\_button\_1" will be created for it using `lv_obj_get_name_resolved`.

**Parametri**

**parent** -- the widget where the search should start

**Ritorna**

the found widget or NULL if not found.

`lv_obj_t *lv_obj_get_child_by_name(const lv_obj_t *parent, const char *name_path)`

Get an object by name. The name can be a path too, for example "main\_container/lv\_button\_1/label". In this case the first part of the name-path should be the direct child of the parent, the second part, should be the direct child of first one, etc.

If the name of a widget was not set a name like "lv\_button\_1" will be created for it using `lv_obj_get_name_resolved`.

**Parametri**

**parent** -- the widget where the search should start

**Ritorna**

the found widget or NULL if not found.

`int32_t lv_obj_get_index(const lv_obj_t *obj)`

Get the index of a child.

**Parametri**

**obj** -- pointer to an object

**Ritorna**

the child index of the object. E.g. 0: the oldest (firstly created child). (-1 if child could not be found or no parent exists)

`int32_t lv_obj_get_index_by_type(const lv_obj_t *obj, const lv_obj_class_t *class_p)`

Get the index of a child. Consider the children only with a given type.

**Parametri**

- **obj** -- pointer to an object
- **class\_p** -- the type of the children to check

**Ritorna**

the child index of the object. E.g. 0: the oldest (firstly created child with the given class). (-1 if child could not be found or no parent exists)

`void lv_obj_tree_walk(lv_obj_t *start_obj, lv_obj_tree_walk_cb_t cb, void *user_data)`

Iterate through all children of any object.

**Parametri**

- **start\_obj** -- start integrating from this object
- **cb** -- call this callback on the objects
- **user\_data** -- pointer to any user related data (will be passed to **cb**)

`void lv_obj_dump_tree(lv_obj_t *start_obj)`

Iterate through all children of any object and print their ID.

**Parametri**

**start\_obj** -- start integrating from this object

## 14.10.19 lv\_refr.h

### Functions

`void lv_refr_now(lv_display_t *disp)`

Redraw the invalidated areas now. Normally the redrawing is periodically executed in `lv_timer_handler` but a long blocking process can prevent the call of `lv_timer_handler`. In this case if the GUI is updated in the process (e.g. progress bar) this function can be called when the screen should be updated.

**Parametri**

**disp** -- pointer to display to refresh. NULL to refresh all displays.

**void lv\_obj\_redraw(lv\_layer\_t \*layer, lv\_obj\_t \*obj)**  
 Redrawn on object and all its children using the passed draw context  
**Parametri**

- **layer** -- pointer to a layer where to draw.
- **obj** -- the start object from the redraw should start

**void lv\_display\_refr\_timer(lv\_timer\_t \*timer)**  
 Called periodically to handle the refreshing  
**Parametri**

- **timer** -- pointer to the timer itself, or NULL

## 14.10.20 lv\_refr\_private.h

### Functions

**void lv\_refr\_init(void)**  
 Initialize the screen refresh subsystem

**void lv\_refr\_deinit(void)**  
 Deinitialize the screen refresh subsystem

**void lv\_inv\_area(lv\_display\_t \*disp, const lv\_area\_t \*area\_p)**  
 Invalidate an area on display to redraw it  
**Parametri**

- **area\_p** -- pointer to area which should be invalidated (NULL: delete the invalidated areas)
- **disp** -- pointer to display where the area should be invalidated (NULL can be used if there is only one display)

**lv\_display\_t \*lv\_refr\_get\_disp\_refreshing(void)**  
 Get the display which is being refreshed  
**Ritorna**

the display being refreshed

**void lv\_refr\_set\_disp\_refreshing(lv\_display\_t \*disp)**  
 Set the display which is being refreshed  
**Parametri**

• **disp** -- the display being refreshed

**lv\_obj\_t \*lv\_refr\_get\_top\_obj(const lv\_area\_t \*area\_p, lv\_obj\_t \*obj)**  
 Search the most top object which fully covers an area  
**Parametri**

- **area\_p** -- pointer to an area
- **obj** -- the first object to start the searching (typically a screen)

**Ritorna**

**void lv\_obj\_refr(lv\_layer\_t \*layer, lv\_obj\_t \*obj)**  
 Render an object to a layer  
**Parametri**

- **layer** -- target drawing layer
- **obj** -- object to render

## 14.11 display

### 14.11.1 lv\_display.h

#### Defines

## LV\_HOR\_RES

The horizontal resolution of the currently active display.

## LV\_VER\_RES

The vertical resolution of the currently active display.

## LV\_DPX\_CALC(dpi, n)

See [lv\\_dpx\(\)](#) and [lv\\_display\\_dpx\(\)](#). Same as Android's DIP. (Different name is chosen to avoid mistype between LV\_DPI and LV\_DIP)

- 40 dip is 40 px on a 160 DPI screen (distance = 1/4 inch).
- 40 dip is 80 px on a 320 DPI screen (distance still = 1/4 inch).

### Vedi anche

<https://stackoverflow.com/questions/2025282/what-is-the-difference-between-px-dip-dp-and-sp>

## LV\_DPX(n)

### Typedef

```
typedef void (*lv_display_flush_cb_t)(lv_display_t *disp, const lv_area_t *area, uint8_t *px_map)
```

```
typedef void (*lv_display_flush_wait_cb_t)(lv_display_t *disp)
```

### Enums

#### enum lv\_display\_rotation\_t

Values:

enumerator **LV\_DISPLAY\_ROTATION\_0**

enumerator **LV\_DISPLAY\_ROTATION\_90**

enumerator **LV\_DISPLAY\_ROTATION\_180**

enumerator **LV\_DISPLAY\_ROTATION\_270**

#### enum lv\_display\_render\_mode\_t

Values:

enumerator **LV\_DISPLAY\_RENDER\_MODE\_PARTIAL**

Use the buffer(s) to render the screen in smaller parts. This way the buffers can be smaller than the display to save RAM. At least 1/10 screen size buffer(s) are recommended.

enumerator **LV\_DISPLAY\_RENDER\_MODE\_DIRECT**

The buffer(s) has to be screen sized and LVGL will render into the correct location of the buffer. This way the buffer always contain the whole image. Only the changed areas will be updated. With 2 buffers the buffers' content are kept in sync automatically and in flush\_cb only address change is required.

### enumerator **LV\_DISPLAY\_RENDER\_MODE\_FULL**

Always redraw the whole screen even if only one pixel has been changed. With 2 buffers in flush\_cb only an address change is required.

enum **lv\_screen\_load\_anim\_t**

*Values:*

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_NONE**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_OVER\_LEFT**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_OVER\_RIGHT**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_OVER\_TOP**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_OVER\_BOTTOM**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_MOVE\_LEFT**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_MOVE\_RIGHT**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_MOVE\_TOP**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_MOVE\_BOTTOM**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_FADE\_IN**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_FADE\_ON**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_FADE\_OUT**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_OUT\_LEFT**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_OUT\_RIGHT**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_OUT\_TOP**

#### enumerator **LV\_SCREEN\_LOAD\_ANIM\_OUT\_BOTTOM**

### Functions

`lv_display_t *lv_display_create(int32_t hor_res, int32_t ver_res)`

Create a new display with the given resolution

#### Parametri

- **hor\_res** -- horizontal resolution in pixels
- **ver\_res** -- vertical resolution in pixels

#### Ritorna

pointer to a display object or NULL on error

**void lv\_display\_delete(lv\_display\_t \*disp)**

Remove a display

**Parametri**

- **disp** -- pointer to display

**void lv\_display\_set\_default(lv\_display\_t \*disp)**

Set a default display. The new screens will be created on it by default.

**Parametri**

- **disp** -- pointer to a display

**lv\_display\_t \*lv\_display\_get\_default(void)**

Get the default display

**Ritorna**

- pointer to the default display

**lv\_display\_t \*lv\_display\_get\_next(lv\_display\_t \*disp)**

Get the next display.

**Parametri**

- **disp** -- pointer to the current display. NULL to initialize.

**Ritorna**

- the next display or NULL if no more. Gives the first display when the parameter is NULL.

**void lv\_display\_set\_resolution(lv\_display\_t \*disp, int32\_t hor\_res, int32\_t ver\_res)**

Sets the resolution of a display. LV\_EVENT\_RESOLUTION\_CHANGED event will be sent. Here the native resolution of the device should be set. If the display will be rotated later with **lv\_display\_set\_rotation** LVGL will swap the hor. and ver. resolution automatically.

**Parametri**

- **disp** -- pointer to a display
- **hor\_res** -- the new horizontal resolution
- **ver\_res** -- the new vertical resolution

**void lv\_display\_set\_physical\_resolution(lv\_display\_t \*disp, int32\_t hor\_res, int32\_t ver\_res)**

It's not mandatory to use the whole display for LVGL, however in some cases physical resolution is important. For example the touchpad still sees whole resolution and the values needs to be converted to the active LVGL display area.

**Parametri**

- **disp** -- pointer to a display
- **hor\_res** -- the new physical horizontal resolution, or -1 to assume it's the same as the normal hor. res.
- **ver\_res** -- the new physical vertical resolution, or -1 to assume it's the same as the normal hor. res.

**void lv\_display\_set\_offset(lv\_display\_t \*disp, int32\_t x, int32\_t y)**

If physical resolution is not the same as the normal resolution the offset of the active display area can be set here.

**Parametri**

- **disp** -- pointer to a display
- **x** -- X offset
- **y** -- Y offset

**void lv\_display\_set\_rotation(lv\_display\_t \*disp, lv\_display\_rotation\_t rotation)**

Set the rotation of this display. LVGL will swap the horizontal and vertical resolutions internally.

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)
- **rotation** -- LV\_DISPLAY\_ROTATION\_0/90/180/270

**void lv\_display\_set\_matrix\_rotation(lv\_display\_t \*disp, bool enable)**

Use matrix rotation for the display. This function is depended on LV\_DRAW\_TRANSFORM\_USE\_MATRIX

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)

- **enable** -- true: enable matrix rotation, false: disable

**void lv\_display\_set\_dpi(lv\_display\_t \*disp, int32\_t dpi)**

Set the DPI (dot per inch) of the display.  $dpi = \sqrt{hor\_res^2 + ver\_res^2} / \text{diagonal}$ "

**Parametri**

- **disp** -- pointer to a display
- **dpi** -- the new DPI

**int32\_t lv\_display\_get\_horizontal\_resolution(const lv\_display\_t \*disp)**

Get the horizontal resolution of a display.

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

the horizontal resolution of the display.

**int32\_t lv\_display\_get\_vertical\_resolution(const lv\_display\_t \*disp)**

Get the vertical resolution of a display

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

the vertical resolution of the display

**int32\_t lv\_display\_get\_original\_horizontal\_resolution(const lv\_display\_t \*disp)**

Get the original horizontal resolution of a display without considering rotation

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

the horizontal resolution of the display.

**int32\_t lv\_display\_get\_original\_vertical\_resolution(const lv\_display\_t \*disp)**

Get the original vertical resolution of a display without considering rotation

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

the vertical resolution of the display

**int32\_t lv\_display\_get\_physical\_horizontal\_resolution(const lv\_display\_t \*disp)**

Get the physical horizontal resolution of a display

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

the physical horizontal resolution of the display

**int32\_t lv\_display\_get\_physical\_vertical\_resolution(const lv\_display\_t \*disp)**

Get the physical vertical resolution of a display

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

the physical vertical resolution of the display

**int32\_t lv\_display\_get\_offset\_x(const lv\_display\_t \*disp)**

Get the horizontal offset from the full / physical display

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

the horizontal offset from the physical display

**int32\_t lv\_display\_get\_offset\_y(const lv\_display\_t \*disp)**

Get the vertical offset from the full / physical display

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

the horizontal offset from the physical display

`lv_display_rotation_t lv_display_get_rotation(lv_display_t *disp)`

Get the current rotation of this display.

**Parametri**

**disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

the current rotation

`bool lv_display_get_matrix_rotation(lv_display_t *disp)`

Get if matrix rotation is enabled for a display or not

**Parametri**

**disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

true: matrix rotation is enabled; false: disabled

`int32_t lv_display_get_dpi(const lv_display_t *disp)`

Get the DPI of the display

**Parametri**

**disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

dpi of the display

`void lv_display_set_buffers(lv_display_t *disp, void *buf1, void *buf2, uint32_t buf_size, lv_display_render_mode_t render_mode)`

Set the buffers for a display, similarly to `lv_display_set_draw_buffers`, but accept the raw buffer pointers. For DIRECT/FULL rendering modes, the buffer size must be at least `hor_res * ver_res * lv_color_format_get_size(lv_display_get_color_format(disp))`

**Parametri**

- **disp** -- pointer to a display
- **buf1** -- first buffer
- **buf2** -- second buffer (can be NULL)
- **buf\_size** -- buffer size in byte
- **render\_mode** -- LV\_DISPLAY\_RENDER\_MODE\_PARTIAL/DIRECT/FULL

`void lv_display_set_buffers_with_stride(lv_display_t *disp, void *buf1, void *buf2, uint32_t buf_size, uint32_t stride, lv_display_render_mode_t render_mode)`

Set the frame buffers for a display, similarly to `lv_display_set_buffers`, but allow for a custom stride as required by a display controller. This allows the frame buffers to have a stride alignment different from the rest of the buffers @param disp pointer to a display @param buf1 first buffer @param buf2 second buffer (can be NULL)

**Parametri**

- **buf\_size** -- buffer size in byte
- **stride** -- buffer stride in bytes
- **render\_mode** -- LV\_DISPLAY\_RENDER\_MODE\_PARTIAL/DIRECT/FULL

`void lv_display_set_draw_buffers(lv_display_t *disp, lv_draw_buf_t *buf1, lv_draw_buf_t *buf2)`

Set the buffers for a display, accept a draw buffer pointer. Normally use `lv_display_set_buffers` is enough for most cases. Use this function when an existing `lv_draw_buf_t` is available.

**Parametri**

- **disp** -- pointer to a display
- **buf1** -- first buffer
- **buf2** -- second buffer (can be NULL)

`void lv_display_set_3rd_draw_buffer(lv_display_t *disp, lv_draw_buf_t *buf3)`

Set the third draw buffer for a display.

**Parametri**

- **disp** -- pointer to a display
- **buf3** -- third buffer

**void lv\_display\_set\_render\_mode(lv\_display\_t \*disp, lv\_display\_render\_mode\_t render\_mode)**

Set display render mode

**Parametri**

- **disp** -- pointer to a display
- **render\_mode** -- LV\_DISPLAY\_RENDER\_MODE\_PARTIAL/DIRECT/FULL

**void lv\_display\_set\_flush\_cb(lv\_display\_t \*disp, lv\_display\_flush\_cb\_t flush\_cb)**

Set the flush callback which will be called to copy the rendered image to the display.

**Parametri**

- **disp** -- pointer to a display
- **flush\_cb** -- the flush callback (px\_map contains the rendered image as raw pixel map and it should be copied to area on the display)

**void lv\_display\_set\_flush\_wait\_cb(lv\_display\_t \*disp, lv\_display\_flush\_wait\_cb\_t wait\_cb)**

Set a callback to be used while LVGL is waiting flushing to be finished. It can do any complex logic to wait, including semaphores, mutexes, polling flags, etc. If not set the disp->flushing flag is used which can be cleared with [lv\\_display\\_flush\\_ready\(\)](#)

**Parametri**

- **disp** -- pointer to a display
- **wait\_cb** -- a callback to call while LVGL is waiting for flush ready. If NULL [lv\\_display\\_flush\\_ready\(\)](#) can be used to signal that flushing is ready.

**void lv\_display\_set\_color\_format(lv\_display\_t \*disp, lv\_color\_format\_t color\_format)**

Set the color format of the display.

**i Nota**

To change the endianness of the rendered image in case of RGB565 format (i.e. swap the 2 bytes) call [lv\\_draw\\_sw\\_rgb565\\_swap](#) in the flush\_cb

**Parametri**

- **disp** -- pointer to a display
- **color\_format** -- Possible values are
  - LV\_COLOR\_FORMAT\_RGB565
  - LV\_COLOR\_FORMAT\_RGB888
  - LV\_COLOR\_FORMAT\_XRGB888
  - LV\_COLOR\_FORMAT\_ARGB888

**lv\_color\_format\_t lv\_display\_get\_color\_format(lv\_display\_t \*disp)**

Get the color format of the display

**Parametri**

- **disp** -- pointer to a display

**Ritorna**

the color format

**void lv\_display\_set\_tile\_cnt(lv\_display\_t \*disp, uint32\_t tile\_cnt)**

Set the number of tiles for parallel rendering.

**Parametri**

- **disp** -- pointer to a display
- **tile\_cnt** -- number of tiles (1 =< tile\_cnt < 256)

`uint32_t lv_display_get_tile_cnt(lv_display_t *disp)`

Get the number of tiles used for parallel rendering

**Parametri**

- **disp** -- pointer to a display

**Ritorna**

number of tiles

`void lv_display_set_antialiasing(lv_display_t *disp, bool en)`

Enable anti-aliasing for the render engine

**Parametri**

- **disp** -- pointer to a display
- **en** -- true/false

`bool lv_display_get_antialiasing(lv_display_t *disp)`

Get if anti-aliasing is enabled for a display or not

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

true/false

`void lv_display_flush_ready(lv_display_t *disp)`

Call from the display driver when the flushing is finished

**Parametri**

- **disp** -- pointer to display whose `flush_cb` was called

`bool lv_display_flush_is_last(lv_display_t *disp)`

Tell if it's the last area of the refreshing process. Can be called from `flush_cb` to execute some special display refreshing if needed when all areas area flushed.

**Parametri**

- **disp** -- pointer to display

**Ritorna**

true: it's the last area to flush; false: there are other areas too which will be refreshed soon

`bool lv_display_is_double_buffered(lv_display_t *disp)`

`lv_obj_t *lv_display_get_screen_active(lv_display_t *disp)`

Return a pointer to the active screen on a display

**Parametri**

- **disp** -- pointer to display which active screen should be get. (NULL to use the default screen)

**Ritorna**

pointer to the active screen object (loaded by `'lv_screen_load()'`)

`lv_obj_t *lv_display_get_screen_prev(lv_display_t *disp)`

Return with a pointer to the previous screen. Only used during screen transitions.

**Parametri**

- **disp** -- pointer to display which previous screen should be get. (NULL to use the default screen)

**Ritorna**

pointer to the previous screen object or NULL if not used now

`lv_obj_t *lv_display_get_layer_top(lv_display_t *disp)`

Return the top layer. The top layer is the same on all screens and it is above the normal screen layer.

**Parametri**

- **disp** -- pointer to display which top layer should be get. (NULL to use the default screen)

**Ritorna**

pointer to the top layer object

`lv_obj_t *lv_display_get_layer_sys(lv_display_t *disp)`

Return the sys. layer. The system layer is the same on all screen and it is above the normal screen and the top layer.

**Parametri**

**disp** -- pointer to display which sys. layer should be retrieved. (NULL to use the default screen)

**Ritorna**

pointer to the sys layer object

```
lv_obj_t *lv_display_get_layer_bottom(lv_display_t *disp)
```

Return the bottom layer. The bottom layer is the same on all screen and it is under the normal screen layer. It's visible only if the screen is transparent.

**Parametri**

**disp** -- pointer to display (NULL to use the default screen)

**Ritorna**

pointer to the bottom layer object

```
lv_obj_t *lv_display_get_screen_by_name(const lv_display_t *disp, const char *screen_name)
```

Get screen by its name on a display. The name should be set by `lv_obj_set_name()` or `lv_obj_set_name_static()`.

**Parametri**

- **disp** -- pointer to a display or NULL to use default display
- **screen\_name** -- name of the screen to get

**Ritorna**

pointer to the screen, or NULL if not found.

```
void lv_screen_load(struct _lv_obj_t *scr)
```

Load a screen on the default display

**Parametri**

**scr** -- pointer to a screen

```
void lv_screen_load_anim(lv_obj_t *scr, lv_screen_load_anim_t anim_type, uint32_t time, uint32_t delay, bool auto_del)
```

Switch screen with animation

**Parametri**

- **scr** -- pointer to the new screen to load
- **anim\_type** -- type of the animation from `lv_screen_load_anim_t`, e.g. `LV_SCREEN_LOAD_ANIM_MOVE_LEFT`
- **time** -- time of the animation
- **delay** -- delay before the transition
- **auto\_del** -- true: automatically delete the old screen

```
lv_obj_t *lv_screen_active(void)
```

Get the active screen of the default display

**Ritorna**

pointer to the active screen

```
lv_obj_t *lv_layer_top(void)
```

Get the top layer of the default display

**Ritorna**

pointer to the top layer

```
lv_obj_t *lv_layer_sys(void)
```

Get the system layer of the default display

**Ritorna**

pointer to the sys layer

```
lv_obj_t *lv_layer_bottom(void)
```

Get the bottom layer of the default display

**Ritorna**

pointer to the bottom layer

```
void lv_display_add_event_cb(lv_display_t *disp, lv_event_cb_t event_cb, lv_event_code_t filter, void *user_data)
```

Add an event handler to the display

**Parametri**

- **disp** -- pointer to a display
- **event\_cb** -- an event callback
- **filter** -- event code to react or **LV\_EVENT\_ALL**
- **user\_data** -- optional user\_data

`uint32_t lv_display_get_event_count(lv_display_t *disp)`

Get the number of event attached to a display

**Parametri**

**disp** -- pointer to a display

**Ritorna**

number of events

`lv_event_dsc_t *lv_display_get_event_dsc(lv_display_t *disp, uint32_t index)`

Get an event descriptor for an event

**Parametri**

- **disp** -- pointer to a display
- **index** -- the index of the event

**Ritorna**

the event descriptor

`bool lv_display_delete_event(lv_display_t *disp, uint32_t index)`

Remove an event

**Parametri**

- **disp** -- pointer to a display
- **index** -- the index of the event to remove

**Ritorna**

true: and event was removed; false: no event was removed

`uint32_t lv_display_remove_event_cb_with_user_data(lv_display_t *disp, lv_event_cb_t event_cb, void *user_data)`

Remove an event\_cb with user\_data

**Parametri**

- **disp** -- pointer to a display
- **event\_cb** -- the event\_cb of the event to remove
- **user\_data** -- user\_data

**Ritorna**

the count of the event removed

`lv_result_t lv_display_send_event(lv_display_t *disp, lv_event_code_t code, void *param)`

Send an event to a display

**Parametri**

- **disp** -- pointer to a display
- **code** -- an event code. **LV\_EVENT\_...**
- **param** -- optional param

**Ritorna**

**LV\_RESULT\_OK**: disp wasn't deleted in the event.

`lv_area_t *lv_event_get_invalidate_area(lv_event_t *e)`

Get the area to be invalidated. Can be used in **LV\_EVENT\_INVALIDATE\_AREA**

**Parametri**

**e** -- pointer to an event

**Ritorna**

the area to invalidated (can be modified as required)

`void lv_display_set_theme(lv_display_t *disp, lv_theme_t *th)`

Set the theme of a display. If there are no user created widgets yet the screens' theme will be updated

**Parametri**

- **disp** -- pointer to a display
- **th** -- pointer to a theme

`lv_theme_t *lv_display_get_theme(lv_display_t *disp)`

Get the theme of a display

**Parametri**

- disp** -- pointer to a display

**Ritorna**

the display's theme (can be NULL)

`uint32_t lv_display_get_inactive_time(const lv_display_t *disp)`

Get elapsed time since last user activity on a display (e.g. click)

**Parametri**

- disp** -- pointer to a display (NULL to get the overall smallest inactivity)

**Ritorna**

elapsed ticks (milliseconds) since the last activity

`void lv_display_trigger_activity(lv_display_t *disp)`

Manually trigger an activity on a display

**Parametri**

- disp** -- pointer to a display (NULL to use the default display)

`void lv_display_enable_invalidation(lv_display_t *disp, bool en)`

Temporarily enable and disable the invalidation of the display.

**Parametri**

- **disp** -- pointer to a display (NULL to use the default display)
- **en** -- true: enable invalidation; false: invalidation

`bool lv_display_is_invalidation_enabled(lv_display_t *disp)`

Get display invalidation is enabled.

**Parametri**

- disp** -- pointer to a display (NULL to use the default display)

**Ritorna**

return true if invalidation is enabled

`lv_timer_t *lv_display_get_refr_timer(lv_display_t *disp)`

Get a pointer to the screen refresher timer to modify its parameters with `lv_timer_...` functions.

**Parametri**

- disp** -- pointer to a display

**Ritorna**

pointer to the display refresher timer. (NULL on error)

`void lv_display_delete_refr_timer(lv_display_t *disp)`

Delete screen refresher timer

**Parametri**

- disp** -- pointer to a display

`bool lv_display_register_vsync_event(lv_display_t *disp, lv_event_cb_t event_cb, void *user_data)`

Register vsync event of a display. `LV_EVENT_VSYNC` event will be sent periodically. Please don't use it in display event listeners, as it may cause memory leaks and illegal access issues.

**Parametri**

- **disp** -- pointer to a display
- **event\_cb** -- an event callback
- **user\_data** -- optional user\_data

`bool lv_display_unregister_vsync_event(lv_display_t *disp, lv_event_cb_t event_cb, void *user_data)`

Unregister vsync event of a display. `LV_EVENT_VSYNC` event won't be sent periodically. Please don't use it in display event listeners, as it may cause memory leaks and illegal access issues.

**Parametri**

- **disp** -- pointer to a display
- **event\_cb** -- an event callback
- **user\_data** -- optional user\_data

`lv_result_t lv_display_send_vsync_event(lv_display_t *disp, void *param)`

Send an vsync event to a display

**Parametri**

- **disp** -- pointer to a display
- **param** -- optional param

**Ritorna**

LV\_RESULT\_OK: disp wasn't deleted in the event.

`void lv_display_set_user_data(lv_display_t *disp, void *user_data)`

`void lv_display_set_driver_data(lv_display_t *disp, void *driver_data)`

`void *lv_display_get_user_data(lv_display_t *disp)`

`void *lv_display_get_driver_data(lv_display_t *disp)`

`lv_draw_buf_t *lv_display_get_buf_active(lv_display_t *disp)`

`void lv_display_rotate_area(lv_display_t *disp, lv_area_t *area)`

Rotate an area in-place according to the display's rotation

**Parametri**

- **disp** -- pointer to a display
- **area** -- pointer to an area to rotate

`uint32_t lv_display_get_draw_buf_size(lv_display_t *disp)`

Get the size of the draw buffers

**Parametri**

**disp** -- pointer to a display

**Ritorna**

the size of the draw buffer in bytes for valid display, 0 otherwise

`uint32_t lv_display_get_invalidate_draw_buf_size(lv_display_t *disp, uint32_t width, uint32_t height)`

Get the size of the invalidated draw buffer. Can be used in the flush callback to get the number of bytes used in the current render buffer.

**Parametri**

- **disp** -- pointer to a display
- **width** -- the width of the invalidated area
- **height** -- the height of the invalidated area

**Ritorna**

the size of the invalidated draw buffer in bytes, not accounting for any preceding palette information for a valid display, 0 otherwise

`int32_t lv_dpx(int32_t n)`

For default display, computes the number of pixels (a distance or size) as if the display had 160 DPI. This allows you to specify 1/160-th fractions of an inch to get real distance on the display that will be consistent regardless of its current DPI. It ensures `lv_dpx(100)`, for example, will have the same physical size regardless to the DPI of the display.

**Parametri**

**n** -- number of 1/160-th-inch units to compute with

**Ritorna**

number of pixels to use to make that distance

`int32_t lv_display_dpx(const lv_display_t *disp, int32_t n)`

For specified display, computes the number of pixels (a distance or size) as if the display had 160 DPI. This allows you to specify 1/160-th fractions of an inch to get real distance on the display that will be consistent regardless of its current DPI. It ensures `lv_dpx(100)`, for example, will have the same physical size regardless to the DPI of the display.

**Parametri**

- **disp** -- pointer to display whose dpi should be considered
- **n** -- number of 1/160-th-inch units to compute with

**Ritorna**

number of pixels to use to make that distance

**14.11.2 lv\_display\_private.h****Defines****LV\_INV\_BUF\_SIZE**

Buffer size for invalid areas

**struct \_lv\_display\_t****Public Members****int32\_t hor\_res**

Horizontal resolution.

**int32\_t ver\_res**

Vertical resolution.

**int32\_t physical\_hor\_res**

Horizontal resolution of the full / physical display. Set to -1 for fullscreen mode.

**int32\_t physical\_ver\_res**

Vertical resolution of the full / physical display. Set to -1 for fullscreen mode.

**int32\_t offset\_x**

Horizontal offset from the full / physical display. Set to 0 for fullscreen mode.

**int32\_t offset\_y**

Vertical offset from the full / physical display. Set to 0 for fullscreen mode.

**uint32\_t dpi**

DPI (dot per inch) of the display. Default value is `LV_DPI_DEF`.

**lv\_draw\_buf\_t \*buf\_1****lv\_draw\_buf\_t \*buf\_2****lv\_draw\_buf\_t \*buf\_3****lv\_draw\_buf\_t \*buf\_act**

Internal, used by the library

***lv\_display\_flush\_cb\_t flush\_cb***

MANDATORY: Write the internal buffer (draw\_buf) to the display. '`lv_display_flush_ready()`' has to be called when finished

*lv\_display\_flush\_wait\_cb\_t* **flush\_wait\_cb**

Used to wait while flushing is ready. It can do any complex logic to wait, including semaphores, mutexes, polling flags, etc. If not set **flushing** flag is used which can be cleared with *lv\_display\_flush\_ready()*

volatile int **flushing**

1: flushing is in progress. (It can't be a bit field because when it's cleared from IRQ Read-Modify-Write issue might occur)

volatile int **flushing\_last**

1: It was the last chunk to flush. (It can't be a bit field because when it's cleared from IRQ Read-Modify-Write issue might occur)

volatile uint32\_t **last\_area**

1: last area is being rendered

volatile uint32\_t **last\_part**

1: last part of the current area is being rendered

*lv\_display\_render\_mode\_t* **render\_mode**uint32\_t **antialiasing**

1: anti-aliasing is enabled on this display.

uint32\_t **tile\_cnt**

Divide the display buffer into these number of tiles

uint32\_t **stride\_is\_auto**

1: The stride of the buffers was not set explicitly.

uint32\_t **rendering\_in\_progress**

1: The current screen rendering is in progress

*lv\_color\_format\_t* **color\_format***lv\_area\_t* **inv\_areas[32]**

Invalidate (marked to redraw) areas

uint8\_t **inv\_area\_joined[32]**uint32\_t **inv\_p**int32\_t **inv\_en\_cnt***lv\_ll\_t* **sync\_areas**

Double buffer sync areas (redrawn during last refresh)

*lv\_draw\_buf\_t* **static\_buf1**

Used when user pass in a raw buffer as display draw buffer

**lv\_draw\_buf\_t static\_buf2**

**lv\_layer\_t \*layer\_head**

void (\***layer\_init**)(lv\_display\_t \*disp, lv\_layer\_t \*layer)

void (\***layer\_deinit**)(lv\_display\_t \*disp, lv\_layer\_t \*layer)

**lv\_obj\_t \*\*screens**

Screens of the display Array of screen objects.

**lv\_obj\_t \*sys\_layer**

 **Vedi anche**

*lv\_display\_get\_layer\_sys*

**lv\_obj\_t \*top\_layer**

 **Vedi anche**

*lv\_display\_get\_layer\_top*

**lv\_obj\_t \*act\_scr**

Currently active screen on this display

**lv\_obj\_t \*bottom\_layer**

 **Vedi anche**

*lv\_display\_get\_layer\_bottom*

**lv\_obj\_t \*prev\_scr**

Previous screen. Used during screen animations

**lv\_obj\_t \*scr\_to\_load**

The screen prepared to load in lv\_screen\_load\_anim

**uint32\_t screen\_cnt**

**uint8\_t draw\_prev\_over\_act**

**uint8\_t del\_prev**

1: Draw previous screen over active screen

**void \*driver\_data**

1: Automatically delete the previous screen when the screen load animation is ready Custom user data

**void \*user\_data**

Custom user data

***lv\_event\_list\_t* event\_list****uint32\_t rotation**

Element of lv\_display\_rotation\_t

**uint32\_t matrix\_rotation**

1: Use matrix for display rotation

**lv\_theme\_t \*theme**

The theme assigned to the screen

**lv\_timer\_t \*refr\_timer**

A timer which periodically checks the dirty areas and refreshes them

**uint32\_t last\_activity\_time**

Last time when there was activity on this display

***lv\_area\_t* refreshed\_area**

The area being refreshed

**uint32\_t vsync\_count****lv\_obj\_t \*perf\_label****lv\_sysmon\_backend\_data\_t perf\_sysmon\_backend****lv\_sysmon\_perf\_info\_t perf\_sysmon\_info****lv\_obj\_t \*mem\_label**

## 14.12 draw

### 14.12.1 lv\_draw.h

#### Defines

**LV\_DRAW\_UNIT\_NONE**

Modified by NXP in 2024

**LV\_DRAW\_UNIT\_IDLE**

The draw unit is idle, new dispatching might be requested to try again

## Enums

enum **lv\_draw\_task\_type\_t**

*Values:*

enumerator **LV\_DRAW\_TASK\_TYPE\_NONE**

enumerator **LV\_DRAW\_TASK\_TYPE\_FILL**

enumerator **LV\_DRAW\_TASK\_TYPE\_BORDER**

enumerator **LV\_DRAW\_TASK\_TYPE\_BOX\_SHADOW**

enumerator **LV\_DRAW\_TASK\_TYPE LETTER**

enumerator **LV\_DRAW\_TASK\_TYPE\_LABEL**

enumerator **LV\_DRAW\_TASK\_TYPE\_IMAGE**

enumerator **LV\_DRAW\_TASK\_TYPE\_LAYER**

enumerator **LV\_DRAW\_TASK\_TYPE\_LINE**

enumerator **LV\_DRAW\_TASK\_TYPE\_ARC**

enumerator **LV\_DRAW\_TASK\_TYPE\_TRIANGLE**

enumerator **LV\_DRAW\_TASK\_TYPE\_MASK\_RECTANGLE**

enumerator **LV\_DRAW\_TASK\_TYPE\_MASK\_BITMAP**

enumerator **LV\_DRAW\_TASK\_TYPE\_VECTOR**

enumerator **LV\_DRAW\_TASK\_TYPE\_3D**

enum **lv\_draw\_task\_state\_t**

*Values:*

enumerator **LV\_DRAW\_TASK\_STATE\_BLOCKED**

Waiting for an other task to be finished. For example in case of **LV\_DRAW\_TASK\_TYPE\_LAYER** (used to blend a layer) is blocked until all the draw tasks of the layer is rendered.

enumerator **LV\_DRAW\_TASK\_STATE\_WAITING**

The draw task is added to the layers list and waits to be rendered.

**enumerator LV\_DRAW\_TASK\_STATE\_QUEUED**

The draw task is added to the command queue of the draw unit. As the queued task are executed in order it's possible to queue multiple draw task (for the same draw unit) even if they are depending on each other. Therefore `lv_draw_get_available_task` and `lv_draw_get_next_available_task` can return draw task for the same draw unit even if a dependent draw task is not finished ready yet.

**enumerator LV\_DRAW\_TASK\_STATE\_IN\_PROGRESS**

The draw task is being rendered. This draw task needs to be finished before `lv_draw_get_available_task` and `lv_draw_get_next_available_task` would return any depending draw tasks.

**enumerator LV\_DRAW\_TASK\_STATE\_FINISHED**

The draw task is rendered. It will be removed from the draw task list of the layer and freed automatically.

**Functions**

**void lv\_draw\_init(void)**

Used internally to initialize the drawing module

**void lv\_draw\_deinit(void)**

Deinitialize the drawing module

**void \*lv\_draw\_create\_unit(size\_t size)**

Allocate a new draw unit with the given size and appends it to the list of draw units

**Parametri**

- **size** -- the size to allocate. E.g. `sizeof(my_draw_unit_t)`, where the first element of `my_draw_unit_t` is `lv_draw_unit_t`.

**lv\_draw\_task\_t \*lv\_draw\_add\_task(lv\_layer\_t \*layer, const lv\_area\_t \*coords, lv\_draw\_task\_type\_t type)**

Add an empty draw task to the draw task list of a layer.

**Parametri**

- **layer** -- pointer to a layer
- **coords** -- the coordinates of the draw task

**Ritorna**

the created draw task which needs to be further configured e.g. by added a draw descriptor

**void lv\_draw\_finalize\_task\_creation(lv\_layer\_t \*layer, lv\_draw\_task\_t \*t)**

Needs to be called when a draw task is created and configured. It will send an event about the new draw task to the widget and assign it to a draw unit.

**Parametri**

- **layer** -- pointer to a layer
- **t** -- pointer to a draw task

**void lv\_draw\_dispatch(void)**

Try dispatching draw tasks to draw units

**bool lv\_draw\_dispatch\_layer(lv\_display\_t \*disp, lv\_layer\_t \*layer)**

Used internally to try dispatching draw tasks of a specific layer

**Parametri**

- **disp** -- pointer to a display on which the dispatching was requested
- **layer** -- pointer to a layer

**Ritorna**

at least one draw task is being rendered (maybe it was taken earlier)

**void lv\_draw\_dispatch\_wait\_for\_request(void)**

Wait for a new dispatch request. It's blocking if `LV_USE_OS == 0` else it yields

**void lv\_draw\_wait\_for\_finish(void)**

Wait for draw finish in case of asynchronous task execution. If `LV_USE_OS == 0` it just return.

**void lv\_draw\_dispatch\_request(void)**

When a draw unit finished a draw task it needs to request dispatching to let LVGL assign a new draw task to it

**uint32\_t lv\_draw\_get\_unit\_count(void)**

Get the total number of draw units.

**lv\_draw\_task\_t \*lv\_draw\_get\_available\_task(lv\_layer\_t \*layer, lv\_draw\_task\_t \*t\_prev, uint8\_t draw\_unit\_id)**

If there is only one draw unit check the first draw task if it's available. If there are multiple draw units call `lv_draw_get_next_available_task` to find a task.

**Parametri**

- **layer** -- the draw layer to search in
- **t\_prev** -- continue searching from this task
- **draw\_unit\_id** -- check the task where `preferred_draw_unit_id` equals this value or `LV_DRAW_UNIT_NONE`

**Ritorna**

an available draw task or NULL if there is not any

**lv\_draw\_task\_t \*lv\_draw\_get\_next\_available\_task(lv\_layer\_t \*layer, lv\_draw\_task\_t \*t\_prev, uint8\_t draw\_unit\_id)**

Find and available draw task

**Parametri**

- **layer** -- the draw layer to search in
- **t\_prev** -- continue searching from this task
- **draw\_unit\_id** -- check the task where `preferred_draw_unit_id` equals this value or `LV_DRAW_UNIT_NONE`

**Ritorna**

an available draw task or NULL if there is not any

**uint32\_t lv\_draw\_get\_dependent\_count(lv\_draw\_task\_t \*t\_check)**

Tell how many draw task are waiting to be drawn on the area of `t_check`. It can be used to determine if a GPU shall combine many draw tasks into one or not. If a lot of tasks are waiting for the current ones it makes sense to draw them one-by-one to not block the dependent tasks' rendering

**Parametri**

`t_check` -- the task whose dependent tasks shall be counted

**Ritorna**

number of tasks depending on `t_check`

**void lv\_layer\_init(lv\_layer\_t \*layer)**

Initialize a layer

**Parametri**

`layer` -- pointer to a layer to initialize

**void lv\_layer\_reset(lv\_layer\_t \*layer)**

Reset the layer to a drawable state

**Parametri**

`layer` -- pointer to a layer to reset

**lv\_layer\_t \*lv\_draw\_layer\_create(lv\_layer\_t \*parent\_layer, lv\_color\_format\_t color\_format, const lv\_area\_t \*area)**

Create (allocate) a new layer on a parent layer

**Parametri**

- **parent\_layer** -- the parent layer to which the layer will be merged when it's rendered
- **color\_format** -- the color format of the layer
- **area** -- the areas of the layer (absolute coordinates)

**Ritorna**

the new target\_layer or NULL on error

```
void lv_draw_layer_init(lv_layer_t *layer, lv_layer_t *parent_layer, lv_color_format_t color_format,
                        const lv_area_t *area)
```

Initialize a layer which is allocated by the user

**Parametri**

- **layer** -- pointer the layer to initialize (its lifetime needs to be managed by the user)
- **parent\_layer** -- the parent layer to which the layer will be merged when it's rendered
- **color\_format** -- the color format of the layer
- **area** -- the areas of the layer (absolute coordinates)

**Ritorna**

the new target\_layer or NULL on error

```
void *lv_draw_layer_alloc_buf(lv_layer_t *layer)
```

Try to allocate a buffer for the layer.

**Parametri**

**layer** -- pointer to a layer

**Ritorna**

pointer to the allocated aligned buffer or NULL on failure

```
void *lv_draw_layer_go_to_xy(lv_layer_t *layer, int32_t x, int32_t y)
```

Got to a pixel at X and Y coordinate on a layer

**Parametri**

- **layer** -- pointer to a layer
- **x** -- the target X coordinate
- **y** -- the target Y coordinate

**Ritorna**

**buf** offset to point to the given X and Y coordinate

```
lv_draw_task_type_t lv_draw_task_get_type(const lv_draw_task_t *t)
```

Get the type of a draw task

**Parametri**

**t** -- the draw task to get the type of

**Ritorna**

the draw task type

```
void *lv_draw_task_get_draw_dsc(const lv_draw_task_t *t)
```

Get the draw descriptor of a draw task

**Parametri**

**t** -- the draw task to get the draw descriptor of

**Ritorna**

a void pointer to the draw descriptor

```
void lv_draw_task_get_area(const lv_draw_task_t *t, lv_area_t *area)
```

Get the draw area of a draw task

**Parametri**

- **t** -- the draw task to get the draw area of
- **area** -- the destination where the draw area will be stored

struct **\_lv\_layer\_t**

**Public Members**

lv\_draw\_buf\_t \***draw\_buf**

Target draw buffer of the layer

**lv\_draw\_task\_t \*draw\_task\_head**

Linked list of draw tasks

**lv\_layer\_t \*parent**

Parent layer

**lv\_layer\_t \*next**

Next layer

**void \*user\_data**

User data

*lv\_area\_t* **buf\_area**

The absolute coordinates of the buffer

*lv\_area\_t* **phy\_clip\_area**

The physical clipping area relative to the display

*lv\_area\_t* **\_clip\_area**

NEVER USE IT DRAW UNITS. USED INTERNALLY DURING DRAW TASK CREATION. The current clip area with absolute coordinates, always the same or smaller than **buf\_area**. Can be set before new draw tasks are added to indicate the clip area of the draw tasks. Therefore *lv\_draw\_add\_task()* always saves it in the new draw task to know the clip area when the draw task was added. During drawing the draw units also sees the saved clip\_area and should use it during drawing. During drawing the layer's clip area shouldn't be used as it might be already changed for other draw tasks.

**int32\_t partial\_y\_offset**

Partial y offset

*lv\_color32\_t* **recolor**

Recolor of the layer

*lv\_color\_format\_t* **color\_format**

The color format of the layer. LV\_COLOR\_FORMAT\_...

**bool all\_tasks\_added**

Flag indicating all tasks are added

*lv\_opa\_t* **opa**

Opacity of the layer

struct **lv\_draw\_dsc\_base\_t**

### Public Members

**lv\_obj\_t \*obj**

The widget for which draw descriptor was created

**uint32\_t part**

The widget part for which draw descriptor was created

**uint32\_t id1**

A widget type specific ID (e.g. table row index). See the docs of the given widget.

**uint32\_t id2**

A widget type specific ID (e.g. table column index). See the docs of the given widget.

**lv\_layer\_t \*layer**

The target layer

**size\_t dsc\_size**

Size of the specific draw descriptor into which this base descriptor is embedded

**void \*user\_data**

Any custom user data

## 14.12.2 lv\_draw\_3d.h

### Functions

**void lv\_draw\_3d\_dsc\_init(lv\_draw\_3d\_dsc\_t \*dsc)**

Initialize a 3D draw descriptor

**Parametri**

**dsc** -- pointer to a draw descriptor

**lv\_draw\_3d\_dsc\_t \*lv\_draw\_task\_get\_3d\_dsc(lv\_draw\_task\_t \*task)**

Try to get a 3D draw descriptor from a draw task.

**Parametri**

**task** -- draw task

**Ritorna**

the task's draw descriptor or NULL if the task is not of type LV\_DRAW\_TASK\_TYPE\_3D

**void lv\_draw\_3d(lv\_layer\_t \*layer, const lv\_draw\_3d\_dsc\_t \*dsc, const lv\_area\_t \*coords)**

Create a 3D draw task

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized *lv\_draw\_3d\_dsc\_t* variable

**struct lv\_draw\_3d\_dsc\_t**

### Public Members

**lv\_draw\_dsc\_base\_t base****lv\_3dtexture\_id\_t tex\_id****lv\_opa\_t opa**

## 14.12.3 lv\_draw\_arc.h

### Functions

**void lv\_draw\_arc\_dsc\_init(*lv\_draw\_arc\_dsc\_t* \*dsc)**

Initialize an arc draw descriptor.

**Parametri**

**dsc** -- pointer to a draw descriptor

***lv\_draw\_arc\_dsc\_t* \*lv\_draw\_task\_get\_arc\_dsc(*lv\_draw\_task\_t* \*task)**

Try to get an arc draw descriptor from a draw task.

**Parametri**

**task** -- draw task

**Ritorna**

the task's draw descriptor or NULL if the task is not of type LV\_DRAW\_TASK\_TYPE\_ARC

**void lv\_draw\_arc(*lv\_layer\_t* \*layer, const *lv\_draw\_arc\_dsc\_t* \*dsc)**

Create an arc draw task.

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor variable

**void lv\_draw\_arc\_get\_area(int32\_t x, int32\_t y, uint16\_t radius, *lv\_value\_precise\_t* start\_angle, *lv\_value\_precise\_t* end\_angle, int32\_t w, bool rounded, *lv\_area\_t* \*area)**

Get an area the should be invalidated when the arcs angle changed between start\_angle and end\_angle

**Parametri**

- **x** -- the x coordinate of the center of the arc
- **y** -- the y coordinate of the center of the arc
- **radius** -- the radius of the arc
- **start\_angle** -- the start angle of the arc (0 deg on the bottom, 90 deg on the right)
- **end\_angle** -- the end angle of the arc
- **w** -- width of the arc
- **rounded** -- true: the arc is rounded
- **area** -- store the area to invalidate here

struct **lv\_draw\_arc\_dsc\_t**

### Public Members

*lv\_draw\_dsc\_base\_t* **base**

*lv\_color\_t* **color**

The color of the arc

int32\_t **width**

The width (thickness) of the arc

*lv\_value\_precise\_t* **start\_angle**

The start angle in 1 degree units (if LV\_USE\_FLOAT is enabled a float number can also be used) 0° is the 3 o'clock position, 90° is the 6 o'clock, etc.

*lv\_value\_precise\_t* **end\_angle**

The end angle, similarly to start\_angle.

*lv\_point\_t* **center**

The center point of the arc.

**const void \*img\_src**

An image source to be used instead of **color**. **NULL** if unused

**uint16\_t radius**

The outer radius of the arc

**lv\_opa\_t opa**

Opacity of the arc in 0...255 range. **LV\_OPA\_TRANSP**, **LV\_OPA\_10**, **LV\_OPA\_20**, .. **LV\_OPA\_COVER** can be used as well

**uint8\_t rounded**

1: Make the arc ends rounded

#### 14.12.4 lv\_draw\_buf.h

##### Defines

**LV\_STRIDE\_AUTO**

Use this value to let LVGL calculate stride automatically

**LV\_DRAW\_BUF\_STRIDE(w, cf)**

Stride alignment for draw buffers. It may vary between different color formats and hardware. Refine it to suit your needs.

**LV\_DRAW\_BUF\_SIZE(w, h, cf)**

Allocate a slightly larger buffer, so we can adjust the start address to meet alignment

**LV\_DRAW\_BUF\_DEFINE\_STATIC(name, \_w, \_h, \_cf)**

Define a static draw buffer with the given width, height, and color format. Stride alignment is set to **LV\_DRAW\_BUF\_STRIDE\_ALIGN**.

For platform that needs special buffer alignment, call **LV\_DRAW\_BUF\_INIT\_STATIC**.

**LV\_DRAW\_BUF\_INIT\_STATIC(name)**

##### Typedef

typedef void \*(\***lv\_draw\_buf\_malloc\_cb**)(size\_t size, *lv\_color\_format\_t* color\_format)

typedef void (\***lv\_draw\_buf\_free\_cb**)(void \*draw\_buf)

typedef void \*(\***lv\_draw\_buf\_align\_cb**)(void \*buf, *lv\_color\_format\_t* color\_format)

typedef void (\***lv\_draw\_buf\_cache\_operation\_cb**)(const **lv\_draw\_buf\_t** \*draw\_buf, const **lv\_area\_t** \*area)

typedef uint32\_t (\***lv\_draw\_buf\_width\_to\_stride\_cb**)(uint32\_t w, *lv\_color\_format\_t* color\_format)

##### Functions

void **lv\_draw\_buf\_init\_with\_default\_handlers**(**lv\_draw\_buf\_handlers\_t** \*handlers)

Initialize the draw buffer with the default handlers.

##### Parametri

**handlers** -- the draw buffer handlers to set

```
void lv_draw_buf_handlers_init(lv_draw_buf_handlers_t *handlers, lv_draw_buf_malloc_cb
                                buf_malloc_cb, lv_draw_buf_free_cb buf_free_cb,
                                lv_draw_buf_align_cb align_pointer_cb,
                                lv_draw_buf_cache_operation_cb invalidate_cache_cb,
                                lv_draw_buf_cache_operation_cb flush_cache_cb,
                                lv_draw_buf_width_to_stride_cb width_to_stride_cb)
```

Initialize the draw buffer with given handlers.

**Parametri**

- **handlers** -- the draw buffer handlers to set
- **buf\_malloc\_cb** -- the callback to allocate memory for the buffer
- **buf\_free\_cb** -- the callback to free memory of the buffer
- **align\_pointer\_cb** -- the callback to align the buffer
- **invalidate\_cache\_cb** -- the callback to invalidate the cache of the buffer
- **flush\_cache\_cb** -- the callback to flush buffer
- **width\_to\_stride\_cb** -- the callback to calculate the stride based on the width and color format

```
lv_draw_buf_handlers_t *lv_draw_buf_get_handlers(void)
```

Get the struct which holds the callbacks for draw buf management. Custom callback can be set on the returned value

**Ritorna**

pointer to the struct of handlers

```
lv_draw_buf_handlers_t *lv_draw_buf_get_font_handlers(void)
```

```
lv_draw_buf_handlers_t *lv_draw_buf_get_image_handlers(void)
```

```
void *lv_draw_buf_align(void *buf, lv_color_format_t color_format)
```

Align the address of a buffer. The buffer needs to be large enough for the real data after alignment

**Parametri**

- **buf** -- the data to align
- **color\_format** -- the color format of the buffer

**Ritorna**

the aligned buffer

```
void *lv_draw_buf_align_ex(const lv_draw_buf_handlers_t *handlers, void *buf, lv_color_format_t
                           color_format)
```

Align the address of a buffer with custom draw buffer handlers. The buffer needs to be large enough for the real data after alignment

**Parametri**

- **handlers** -- the draw buffer handlers
- **buf** -- the data to align
- **color\_format** -- the color format of the buffer

**Ritorna**

the aligned buffer

```
void lv_draw_buf_invalidate_cache(const lv_draw_buf_t *draw_buf, const lv_area_t *area)
```

Invalidate the cache of the buffer

**Parametri**

- **draw\_buf** -- the draw buffer needs to be invalidated
- **area** -- the area to invalidate in the buffer, use NULL to invalidate the whole draw buffer address range

```
void lv_draw_buf_flush_cache(const lv_draw_buf_t *draw_buf, const lv_area_t *area)
```

Flush the cache of the buffer

**Parametri**

- **draw\_buf** -- the draw buffer needs to be flushed
- **area** -- the area to flush in the buffer, use NULL to flush the whole draw buffer address range

```
uint32_t lv_draw_buf_width_to_stride(uint32_t w, lv_color_format_t color_format)
```

Calculate the stride in bytes based on a width and color format

**Parametri**

- **w** -- the width in pixels
- **color\_format** -- the color format

**Ritorna**

the stride in bytes

```
uint32_t lv_draw_buf_width_to_stride_ex(const lv_draw_buf_handlers_t *handlers, uint32_t w,
                                         lv_color_format_t color_format)
```

Calculate the stride in bytes based on a width and color format

**Parametri**

- **handlers** -- the draw buffer handlers
- **w** -- the width in pixels
- **color\_format** -- the color format

**Ritorna**

the stride in bytes

```
void lv_draw_buf_clear(lv_draw_buf_t *draw_buf, const lv_area_t *a)
```

Clear an area on the buffer

**Parametri**

- **draw\_buf** -- pointer to draw buffer
- **a** -- the area to clear, or NULL to clear the whole buffer

```
void lv_draw_buf_copy(lv_draw_buf_t *dest, const lv_area_t *dest_area, const lv_draw_buf_t *src, const
                      lv_area_t *src_area)
```

Copy an area from a buffer to another

**i Nota**

**dest\_area** and **src\_area** should have the same width and height

**i Nota**

**dest** and **src** should have same color format. Color converting is not supported now.

**Parametri**

- **dest** -- pointer to the destination draw buffer
- **dest\_area** -- the area to copy from the destination buffer, if NULL, use the whole buffer
- **src** -- pointer to the source draw buffer
- **src\_area** -- the area to copy from the destination buffer, if NULL, use the whole buffer

```
lv_draw_buf_t*lv_draw_buf_create(uint32_t w, uint32_t h, lv_color_format_t cf, uint32_t stride)
```

Note: Eventually, `lv_draw_buf_malloc/free` will be kept as private. For now, we use `create` to distinguish with `malloc`.

Create an draw buf by allocating struct for `lv_draw_buf_t` and allocating a buffer for it that meets specified requirements.

**Parametri**

- **w** -- the buffer width in pixels
- **h** -- the buffer height in pixels
- **cf** -- the color format for image
- **stride** -- the stride in bytes for image. Use 0 for automatic calculation based on `w`, `cf`, and global stride alignment configuration.

```
lv_draw_buf_t *lv_draw_buf_create_ex(const lv_draw_buf_handlers_t *handlers, uint32_t w, uint32_t h,
                                     lv_color_format_t cf, uint32_t stride)
```

Note: Eventually, lv\_draw\_buf\_malloc/free will be kept as private. For now, we use `create` to distinguish with malloc.

Create an draw buf by allocating struct for `lv_draw_buf_t` and allocating a buffer for it that meets specified requirements.

#### Parametri

- **handlers** -- the draw buffer handlers
- **w** -- the buffer width in pixels
- **h** -- the buffer height in pixels
- **cf** -- the color format for image
- **stride** -- the stride in bytes for image. Use 0 for automatic calculation based on w, cf, and global stride alignment configuration.

```
lv_draw_buf_t *lv_draw_buf_dup(const lv_draw_buf_t *draw_buf)
```

Duplicate a draw buf with same image size, stride and color format. Copy the image data too.

#### Parametri

- **draw\_buf** -- the draw buf to duplicate

#### Ritorna

the duplicated draw buf on success, NULL if failed

```
lv_draw_buf_t *lv_draw_buf_dup_ex(const lv_draw_buf_handlers_t *handlers, const lv_draw_buf_t *
                                  *draw_buf)
```

Duplicate a draw buf with same image size, stride and color format. Copy the image data too.

#### Parametri

- **handlers** -- the draw buffer handlers
- **draw\_buf** -- the draw buf to duplicate

#### Ritorna

the duplicated draw buf on success, NULL if failed

```
lv_result_t lv_draw_buf_init(lv_draw_buf_t *draw_buf, uint32_t w, uint32_t h, lv_color_format_t cf,
                             uint32_t stride, void *data, uint32_t data_size)
```

Initialize a draw buf with the given buffer and parameters. Clear draw buffer flag to zero.

#### Parametri

- **draw\_buf** -- the draw buf to initialize
- **w** -- the buffer width in pixels
- **h** -- the buffer height in pixels
- **cf** -- the color format
- **stride** -- the stride in bytes. Use 0 for automatic calculation
- **data** -- the buffer used for drawing. Unaligned **data** will be aligned internally
- **data\_size** -- the size of the buffer in bytes

#### Ritorna

return LV\_RESULT\_OK on success, LV\_RESULT\_INVALID otherwise

```
lv_draw_buf_t *lv_draw_buf_reshape(lv_draw_buf_t *draw_buf, lv_color_format_t cf, uint32_t w,
                                   uint32_t h, uint32_t stride)
```

Keep using the existing memory, reshape the draw buffer to the given width and height. Return NULL if `data_size` is smaller than the required size.

#### Parametri

- **draw\_buf** -- pointer to a draw buffer
- **cf** -- the new color format, use 0 or LV\_COLOR\_FORMAT\_UNKNOWN to keep using the original color format.
- **w** -- the new width in pixels
- **h** -- the new height in pixels
- **stride** -- the stride in bytes for image. Use 0 for automatic calculation.

```
void lv_draw_buf_destroy(lv_draw_buf_t *draw_buf)
```

Destroy a draw buf by freeing the actual buffer if it's marked as `LV_IMAGE_FLAGS_ALLOCATED` in header. Then free the `lv_draw_buf_t` struct.

**Parametri**

- **draw\_buf** -- the draw buffer to destroy

`void *lv_draw_buf_goto_xy(const lv_draw_buf_t *buf, uint32_t x, uint32_t y)`

Return pointer to the buffer at the given coordinates

`lv_result_t lv_draw_buf_adjust_stride(lv_draw_buf_t *src, uint32_t stride)`

Adjust the stride of a draw buf in place.

**Parametri**

- **src** -- pointer to a draw buffer
- **stride** -- the new stride in bytes for image. Use LV\_STRIDE\_AUTO for automatic calculation.

**Ritorna**

LV\_RESULT\_OK: success or LV\_RESULT\_INVALID: failed

`lv_result_t lv_draw_buf_premultiply(lv_draw_buf_t *draw_buf)`

Premultiply draw buffer color with alpha channel. If it's already premultiplied, return directly. Only color formats with alpha channel will be processed.

**Ritorna**

LV\_RESULT\_OK: premultiply success

`bool lv_draw_buf_has_flag(const lv_draw_buf_t *draw_buf, lv_image_flags_t flag)`

`void lv_draw_buf_set_flag(lv_draw_buf_t *draw_buf, lv_image_flags_t flag)`

`void lv_draw_buf_clear_flag(lv_draw_buf_t *draw_buf, lv_image_flags_t flag)`

`lv_result_t lv_draw_buf_from_image(lv_draw_buf_t *buf, const lv_image_dsc_t *img)`

As of now, draw buf share same definition as `lv_image_dsc_t`. And is interchangeable with `lv_image_desc_t`.

`void lv_draw_buf_to_image(const lv_draw_buf_t *buf, lv_image_dsc_t *img)`

`void lv_draw_buf_set_palette(lv_draw_buf_t *draw_buf, uint8_t index, lv_color32_t color)`

Set the palette color of an indexed image. Valid only for LV\_COLOR\_FORMAT\_I1/2/4/8

**Parametri**

- **draw\_buf** -- pointer to an image descriptor
- **index** -- the palette color to set:
  - for LV\_COLOR\_FORMAT\_I1: 0..1
  - for LV\_COLOR\_FORMAT\_I2: 0..3
  - for LV\_COLOR\_FORMAT\_I4: 0..15
  - for LV\_COLOR\_FORMAT\_I8: 0..255
- **color** -- the color to set in `lv_color32_t` format

`void lv_image_buf_set_palette(lv_image_dsc_t *dsc, uint8_t id, lv_color32_t c)`

*Deprecated:*

Use `lv_draw_buf_set_palette` instead.

`void lv_image_buf_free(lv_image_dsc_t *dsc)`

*Deprecated:*

Use `lv_draw_buffer_create/destroy` instead. Free the data pointer and dsc struct of an image.

`struct _lv_draw_buf_t`

**Public Members*****lv\_image\_header\_t* header****uint32\_t data\_size**

Total buf size in bytes

**uint8\_t \*data****void \*unaligned\_data**Unaligned address of **data**, used internally by lvgl**const lv\_draw\_buf\_handlers\_t \*handlers**

draw buffer alloc/free ops.

## 14.12.5 lv\_draw\_buf\_private.h

**Functions****void lv\_draw\_buf\_init\_handlers(void)**

Called internally to initialize the draw\_buf\_handlers in lv\_global

**struct \_lv\_draw\_buf\_handlers\_t****Public Members*****lv\_draw\_buf\_malloc\_cb* buf\_malloc\_cb*****lv\_draw\_buf\_free\_cb* buf\_free\_cb*****lv\_draw\_buf\_align\_cb* align\_pointer\_cb*****lv\_draw\_buf\_cache\_operation\_cb* invalidate\_cache\_cb*****lv\_draw\_buf\_cache\_operation\_cb* flush\_cache\_cb*****lv\_draw\_buf\_width\_to\_stride\_cb* width\_to\_stride\_cb**

## 14.12.6 lv\_draw\_image.h

**TypeDef**

```
typedef void (*lv_draw_image_core_cb)(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const
lv_image_decoder_dsc_t *decoder_dsc, lv_draw_image_sup_t *sup, const lv_area_t *img_coords, const lv_area_t
*clipped_img_area)
```

Perfom the actual rendering of a decoded image

**Param t**

pointer to a draw task

**Param draw\_dsc**

the draw descriptor of the image

**Param decoder\_dsc**

pointer to the decoded image's descriptor

**Param sup**

supplementary data

**Param img\_coords**

the absolute coordinates of the image

**Param clipped\_img\_area**

the absolute clip coordinates

**Functions**

**void lv\_draw\_image\_dsc\_init(lv\_draw\_image\_dsc\_t \*dsc)**

Initialize an image draw descriptor.

**Parametri**

**dsc** -- pointer to a draw descriptor

**lv\_draw\_image\_dsc\_t \*lv\_draw\_task\_get\_image\_dsc(lv\_draw\_task\_t \*task)**

Try to get an image draw descriptor from a draw task.

**Parametri**

**task** -- draw task

**Ritorna**

the task's draw descriptor or NULL if the task is not of type LV\_DRAW\_TASK\_TYPE\_IMAGE

**void lv\_draw\_image(lv\_layer\_t \*layer, const lv\_draw\_image\_dsc\_t \*dsc, const lv\_area\_t \*coords)**

Create an image draw task

**Nota**

**coords** can be small than the real image area (if only a part of the image is rendered) or can be larger (in case of tiled images).

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor
- **coords** -- the coordinates of the image

**void lv\_draw\_layer(lv\_layer\_t \*layer, const lv\_draw\_image\_dsc\_t \*dsc, const lv\_area\_t \*coords)**

Create a draw task to blend a layer to another layer

**Nota**

**coords** can be small than the total widget area from which the layer is created (if only a part of the widget was rendered to a layer)

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor. **src** must be set to the layer to blend
- **coords** -- the coordinates of the layer.

**lv\_image\_src\_t lv\_image\_src\_get\_type(const void \*src)**

Get the type of an image source

**Parametri**

**src** -- pointer to an image source:

- pointer to an 'lv\_image\_t' variable (image stored internally and compiled into the code)
- a path to a file (e.g. "S:/folder/image.bin")
- or a symbol (e.g. LV\_SYMBOL\_CLOSE)

**Ritorna**

type of the image source LV\_IMAGE\_SRC\_VARIABLE/FILE/SYMBOL/UNKNOWN

struct **\_lv\_draw\_image\_dsc\_t**

**Public Members**

*lv\_draw\_dsc\_base\_t* **base**

const void \***src**

The image source: pointer to *lv\_image\_dsc\_t* or a path to a file

*lv\_image\_header\_t* **header**

The header of the image. Initialized internally in *lv\_draw\_image*

int32\_t **clip\_radius**

Clip the corner of the image with this radius. Use LV\_RADIUS\_CIRCLE for max. radius

int32\_t **rotation**

The rotation of the image in 0.1 degree unit. E.g. 234 means 23.4°

int32\_t **scale\_x**

Horizontal scale (zoom) of the image. 256 (LV\_SCALE\_NONE): means no zoom, 512 double size, 128 half size.

int32\_t **scale\_y**

Same as **scale\_x** but vertically

int32\_t **skew\_x**

Parallelogram like transformation of the image horizontally in 0.1 degree unit. E.g. 456 means 45.6°.

int32\_t **skew\_y**

Same as **skew\_x** but vertically

*lv\_point\_t* **pivot**

The pivot point of transformation (scale and rotation). 0;0 is the top left corner of the image. Can be outside of the image too.

*lv\_color\_t* **recolor**

Mix this color to the images. In case of LV\_COLOR\_FORMAT\_A8 it will be the color of the visible pixels

*lv\_opa\_t* **recolor\_opa**

The intensity of recoloring. 0 means, no recolor, 255 means full cover (transparent pixels remain transparent)

*lv\_opa\_t* **opa**

Opacity in 0...255 range. LV\_OPA\_TRANSP, LV\_OPA\_10, LV\_OPA\_20, .. LV\_OPA\_COVER can be used as well

***lv\_blend\_mode\_t blend\_mode***

Describes how to blend the pixels of the image to the background. See [\*lv\\_blend\\_mode\\_t\*](#) for more details.

***uint16\_t antialias***

1: perform the transformation with anti-aliasing

***uint16\_t tile***

If the image is smaller than the `image_area` field of `lv_draw_image_dsc_t` tile the image (repeat is both horizontally and vertically) to fill the `image_area` area

***lv\_draw\_image\_sup\_t \*sup***

Used internally to store some information about the palette or the color of A8 images

***lv\_area\_t image\_area***

Used to indicate the entire original, non-clipped area where the image is to be drawn. This is important for:

- i. Layer rendering, where it might happen that only a smaller area of the layer is rendered and e.g. `clip_radius` needs to know what the original image was.
- ii. Tiled images, where the target draw area is larger than the image to be tiled.

***const lv\_image\_dsc\_t \*bitmap\_mask\_src***

Pointer to an A8 or L8 image descriptor to mask the image with. The mask is always center aligned.

## 14.12.7 `lv_draw_image_private.h`

### Functions

```
void lv_draw_image_normal_helper(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const
                                lv_area_t *coords, lv_draw_image_core_cb draw_core_cb)
```

Can be used by draw units to handle the decoding and prepare everything for the actual image rendering

**Parametri**

- **t** -- pointer to a draw task
- **draw\_dsc** -- the draw descriptor of the image
- **coords** -- the absolute coordinates of the image
- **draw\_core\_cb** -- a callback to perform the actual rendering

```
void lv_draw_image_tiled_helper(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const
                                lv_area_t *coords, lv_draw_image_core_cb draw_core_cb)
```

Can be used by draw units for TILED images to handle the decoding and prepare everything for the actual image rendering

**Parametri**

- **t** -- pointer to a draw task
- **draw\_dsc** -- the draw descriptor of the image
- **coords** -- the absolute coordinates of the image
- **draw\_core\_cb** -- a callback to perform the actual rendering

```
void lv_image_buf_get_transformed_area(lv_area_t *res, int32_t w, int32_t h, int32_t angle,
                                         uint16_t scale_x, uint16_t scale_y, const lv_point_t
                                         *pivot)
```

Get the area of a rectangle if its rotated and scaled

**Parametri**

- **res** -- store the coordinates here
- **w** -- width of the rectangle to transform
- **h** -- height of the rectangle to transform

- **angle** -- angle of rotation
- **scale\_x** -- zoom in x direction, (256 no zoom)
- **scale\_y** -- zoom in y direction, (256 no zoom)
- **pivot** -- x,y pivot coordinates of rotation

```
struct _lv_draw_image_sup_t
```

#### Public Members

*lv\_color\_t* **alpha\_color**

const *lv\_color32\_t* \***palette**

uint32\_t **palette\_size**

### 14.12.8 lv\_draw\_label.h

#### Defines

**LV\_DRAW\_LABEL\_NO\_TXT\_SEL**

#### Typedef

typedef void (\***lv\_draw\_glyph\_cb\_t**)(lv\_draw\_task\_t \*t, lv\_draw\_glyph\_dsc\_t \*dsc, *lv\_draw\_fill\_dsc\_t* \*fill\_dsc, const *lv\_area\_t* \*fill\_area)

Passed as a parameter to **lv\_draw\_label\_iterate\_characters** to draw the characters one by one

##### Param t

pointer to a draw task

##### Param dsc

pointer to **lv\_draw\_glyph\_dsc\_t** to describe the character to draw if NULL don't draw character

##### Param fill\_dsc

pointer to a fill descriptor to draw a background for the character or underline or strike through if NULL do not fill anything

##### Param fill\_area

the area to fill if NULL do not fill anything

#### Functions

void **lv\_draw\_letter\_dsc\_init**(*lv\_draw\_letter\_dsc\_t* \*dsc)

void **lv\_draw\_label\_dsc\_init**(*lv\_draw\_label\_dsc\_t* \*dsc)

Initialize a label draw descriptor

##### Parametri

**dsc** -- pointer to a draw descriptor

*lv\_draw\_label\_dsc\_t* \***lv\_draw\_task\_get\_label\_dsc**(lv\_draw\_task\_t \*task)

Try to get a label draw descriptor from a draw task.

##### Parametri

**task** -- draw task

##### Ritorna

the task's draw descriptor or NULL if the task is not of type LV\_DRAW\_TASK\_TYPE\_LABEL

---

```
void lv_draw_glyph_dsc_init(lv_draw_glyph_dsc_t *dsc)
```

Initialize a glyph draw descriptor. Used internally.

**Parametri**

- **dsc** -- pointer to a draw descriptor

```
void lv_draw_label(lv_layer_t *layer, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords)
```

Create a draw task to render a text

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to draw descriptor
- **coords** -- coordinates of the character

```
void lv_draw_character(lv_layer_t *layer, lv_draw_label_dsc_t *dsc, const lv_point_t *point, uint32_t unicode_letter)
```

Create a draw task to render a single character

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to draw descriptor
- **point** -- position of the label
- **unicode\_letter** -- the letter to draw

```
void lv_draw_letter(lv_layer_t *layer, lv_draw_letter_dsc_t *dsc, const lv_point_t *point)
```

Draw a single letter

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to draw descriptor
- **point** -- position of the label

```
void lv_draw_label_iterate_characters(lv_draw_task_t *t, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords, lv_draw_glyph_cb_t cb)
```

Should be used during rendering the characters to get the position and other parameters of the characters

**Parametri**

- **t** -- pointer to a draw task
- **dsc** -- pointer to draw descriptor
- **coords** -- coordinates of the label
- **cb** -- a callback to call to draw each glyphs one by one

```
void lv_draw_unit_draw_letter(lv_draw_task_t *t, lv_draw_glyph_dsc_t *dsc, const lv_point_t *pos, const lv_font_t *font, uint32_t letter, lv_draw_glyph_cb_t cb)
```

Draw a single letter using the provided draw unit, glyph descriptor, position, font, and callback.

This function is responsible for rendering a single character from a text string, applying the necessary styling described by the glyph descriptor (**dsc**). It handles the retrieval of the glyph's description, checks its visibility within the clipping area, and invokes the callback (**cb**) to render the glyph at the specified position (**pos**) using the given font (**font**).

**Parametri**

- **t** -- Pointer to the drawing task.
- **dsc** -- Pointer to the descriptor containing styling for the glyph to be drawn.
- **pos** -- Pointer to the point coordinates where the letter should be drawn.
- **font** -- Pointer to the font containing the glyph.
- **letter** -- The Unicode code point of the letter to be drawn.
- **cb** -- Callback function to execute the actual rendering of the glyph.

---

```
struct lv_draw_label_dsc_t
```

## Public Members

*lv\_draw\_dsc\_base\_t* **base**

const char \***text**

The text to draw

*lv\_point\_t* **text\_size**

The size of the text

const lv\_font\_t \***font**

The font to use. Fallback fonts are also handled.

*lv\_color\_t* **color**

Color of the text

int32\_t **line\_space**

Extra space between the lines

int32\_t **letter\_space**

Extra space between the characters

int32\_t **ofs\_x**

Offset the text with this value horizontally

int32\_t **ofs\_y**

Offset the text with this value vertically

int32\_t **rotation**

Rotation of the letters in 0.1 degree unit

uint32\_t **sel\_start**

The first characters index for selection (not byte index). LV\_DRAW\_LABEL\_NO\_TXT\_SEL for no selection

uint32\_t **sel\_end**

The last characters's index for selection (not byte index). LV\_DRAW\_LABEL\_NO\_TXT\_SEL for no selection

*lv\_color\_t* **sel\_color**

Color of the selected characters

*lv\_color\_t* **sel\_bg\_color**

Background color of the selected characters

uint32\_t **text\_length**

The number of characters to render. 0: means render until reaching the \0 termination.

*lv\_text\_align\_t* **align**

The alignment of the text LV\_TEXT\_ALIGN\_LEFT/RIGHT/CENTER

*lv\_base\_dir\_t* **bidi\_dir**

The base direction. Used when type setting Right-to-left (e.g. Arabic) texts

*lv\_opa\_t* **opa**

Opacity of the text in 0...255 range. LV\_OPA\_TRANSP, LV\_OPA\_10, LV\_OPA\_20, .. LV\_OPA\_COVER can be used as well

*lv\_opa\_t* **outline\_stroke\_opa**

Letter outline stroke opacity

*lv\_text\_decor\_t* **decor**

Text decoration, e.g. underline

*lv\_text\_flag\_t* **flag**

Some flags to control type setting

*uint8\_t* **text\_local**

1: malloc a buffer and copy **text** there. 0: **text** will be valid during rendering.

*uint8\_t* **text\_static**

Indicate that the text is constant and its pointer can be safely saved e.g. in a cache.

*uint8\_t* **has\_bided**

1: already executed lv\_bidi\_process\_paragraph. 0: has not been executed lv\_bidi\_process\_paragraph.

*lv\_draw\_label\_hint\_t* \***hint**

Pointer to an externally stored struct where some data can be cached to speed up rendering

*lv\_color\_t* **outline\_stroke\_color***int32\_t* **outline\_stroke\_width**struct **lv\_draw\_letter\_dsc\_t****Public Members***lv\_draw\_dsc\_base\_t* **base***uint32\_t* **unicode**const *lv\_font\_t* \***font***lv\_color\_t* **color***int32\_t* **rotation***int32\_t* **scale\_x**

```

int32_t scale_y

int32_t skew_x

int32_t skew_y

lv_point_t pivot

lv_opa_t opa

lv_text_decor_t decor

lv_blend_mode_t blend_mode

lv_opa_t outline_stroke_opa

int32_t outline_stroke_width

lv_color_t outline_stroke_color

```

## 14.12.9 lv\_draw\_label\_private.h

### struct \_lv\_draw\_label\_hint\_t

#include <lv\_draw\_label\_private.h> Store some info to speed up drawing of very large texts It takes a lot of time to get the first visible character because all the previous characters needs to be checked to calculate the positions. This structure stores an earlier (e.g. at -1000 px) coordinate and the index of that line. Therefore the calculations can start from here.

#### Public Members

##### int32\_t **line\_start**

Index of the line at y coordinate

##### int32\_t **y**

Give the y coordinate of the first letter at **line\_start** index. Relative to the label's coordinates

##### int32\_t **coord\_y**

The 'y1' coordinate of the label when the hint was saved. Used to invalidate the hint if the label has moved too much.

### struct \_lv\_draw\_glyph\_dsc\_t

#### Public Members

##### const void \***glyph\_data**

Depends on **format** field, it could be image source or draw buf of bitmap or vector data.

```
lv_font_glyph_format_t format  

const lv_area_t *letter_coords  

const lv_area_t *bg_coords  

lv_font_glyph_dsc_t *g  

lv_color_t color  

lv_opa_t opa  

lv_color_t outline_stroke_color  

lv_opa_t outline_stroke_opa  

int32_t outline_stroke_width  

int32_t rotation
```

***lv\_point\_t* **pivot****

Rotation pivot point associated with total glyph including line\_height

***lv\_draw\_buf\_t* \***\_draw\_buf****

a shared draw buf for get\_bitmap, do not use it directly, use glyph\_data instead

## 14.12.10 lv\_draw\_line.h

### Functions

**void *lv\_draw\_line\_dsc\_init*(*lv\_draw\_line\_dsc\_t* \**dsc*)**

Initialize a line draw descriptor

**Parametri**

**dsc** -- pointer to a draw descriptor

***lv\_draw\_line\_dsc\_t* \***lv\_draw\_task\_get\_line\_dsc**(*lv\_draw\_task\_t* \**task*)**

Try to get a line draw descriptor from a draw task.

**Parametri**

**task** -- draw task

**Ritorna**

the task's draw descriptor or NULL if the task is not of type LV\_DRAW\_TASK\_TYPE\_LINE

**void *lv\_draw\_line*(*lv\_layer\_t* \**layer*, **const** *lv\_draw\_line\_dsc\_t* \**dsc*)**

Create a line draw task

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized *lv\_draw\_line\_dsc\_t* variable

**struct *lv\_draw\_line\_dsc\_t***

**Public Members*****lv\_draw\_dsc\_base\_t* base*****lv\_point\_precise\_t* p1**

The first point of the line. If LV\_USE\_FLOAT is enabled float number can be also used

***lv\_point\_precise\_t* p2**

The second point of the line. If LV\_USE\_FLOAT is enabled float number can be also used

***lv\_color\_t* color**

The color of the line

***int32\_t* width**

The width (thickness) of the line

***int32\_t* dash\_width**

The length of a dash (0: don't dash)

***int32\_t* dash\_gap**

The length of the gaps between dashes (0: don't dash)

***lv\_opa\_t* opa**Opacity of the line in 0...255 range. LV\_OPA\_TRANSP, LV\_OPA\_10, LV\_OPA\_20, ..  
LV\_OPA\_COVER can be used as well***uint8\_t* round\_start**

Make the line start rounded

***uint8\_t* round\_end**

Make the line end rounded

***uint8\_t* raw\_end**

1: Do not bother with line ending (if it's not visible for any reason)

## 14.12.11 lv\_draw\_mask.h

**Functions****void *lv\_draw\_mask\_rect\_dsc\_init*(*lv\_draw\_mask\_rect\_dsc\_t* \*dsc)**

Initialize a rectangle mask draw descriptor.

**Parametri****dsc** -- pointer to a draw descriptor***lv\_draw\_mask\_rect\_dsc\_t* \*lv\_draw\_task\_get\_mask\_rect\_dsc(*lv\_draw\_task\_t* \*task)**

Try to get a rectangle mask draw descriptor from a draw task.

**Parametri****task** -- draw task**Ritorna**

the task's draw descriptor or NULL if the task is not of type LV\_DRAW\_TASK\_TYPE\_MASK\_RECTANGLE

---

```
void lv_draw_mask_rect(lv_layer_t *layer, const lv_draw_mask_rect_dsc_t *dsc)
```

Create a draw task to mask a rectangle from the buffer

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to a draw descriptor

### 14.12.12 lv\_draw\_mask\_private.h

```
struct _lv_draw_mask_rect_dsc_t
```

**Public Members**

*lv\_draw\_dsc\_base\_t* **base**

*lv\_area\_t* **area**

The area t mask.

**int32\_t radius**

The radius of masking

**uint32\_t keep\_outside**

0: clear the content out of the **area**. 1: don't touch the area out of **area**

### 14.12.13 lv\_draw\_private.h

```
struct _lv_draw_task_t
```

#include <lv\_draw\_private.h> Modified by NXP in 2024

**Public Members**

*lv\_draw\_task\_t* \***next**

*lv\_draw\_task\_type\_t* **type**

*lv\_area\_t* **area**

The area where to draw

*lv\_area\_t* **\_real\_area**

The real draw area. E.g. for shadow, outline, or transformed images it's different from **area**

*lv\_area\_t* **clip\_area\_original**

The original area which is updated

*lv\_area\_t* **clip\_area**

The clip area of the layer is saved here when the draw task is created. As the clip area of the layer can be changed as new draw tasks are added its current value needs to be saved. Therefore during drawing the layer's clip area shouldn't be used as it might be already changed for other draw tasks.

*lv\_layer\_t*\***target\_layer**

```
lv_draw_unit_t *draw_unit  

volatile int state  

void *draw_dsc  

    int instead of lv_draw_task_state_t to be sure its atomic  

lv_opa_t opa  

    Opacity of the layer  

uint8_t preferred_draw_unit_id  

    The ID of the draw_unit which should take this task  

uint8_t preference_score  

    Set to which extent preferred_draw_unit_id is good at this task. 80: means 20% better (faster) than software rendering 100: the default value 110: means 10% worse (slower) than software rendering
```

## struct \_lv\_draw\_mask\_t

### Public Members

void \***user\_data**

## struct \_lv\_draw\_unit\_t

### Public Members

lv\_draw\_unit\_t \***next**

const char \***name**

Name and ID of the draw unit, for debugging purposes only.

int32\_t **idx**

int32\_t (\***dispatch\_cb**)(lv\_draw\_unit\_t \*draw\_unit, lv\_layer\_t \*layer)

Called to try to assign a draw task to itself. *lv\_draw\_get\_next\_available\_task* can be used to get an independent draw task. A draw task should be assign only if the draw unit can draw it too

#### Param draw\_unit

pointer to the draw unit

#### Param layer

pointer to a layer on which the draw task should be drawn

#### Return

>=0: The number of taken draw task: 0 means the task has not yet been completed.

1 means a new task has been accepted. -1: The draw unit wanted to work on a task but couldn't do that due to some errors (e.g. out of memory). It signals that LVGL should call the dispatcher later again to let draw unit try to start the rendering again.

int32\_t (\***evaluate\_cb**)(lv\_draw\_unit\_t \*draw\_unit, lv\_draw\_task\_t \*task)

#### Param draw\_unit

#### Param task

**Return**

```
int32_t (*wait_for_finish_cb)(lv_draw_unit_t *draw_unit)
```

Called to signal the unit to complete all tasks in order to return their ready status. This callback can be implemented in case of asynchronous task processing. Below is an example to show the difference between synchronous and asynchronous:

Synchronous: LVGL thread DRAW thread HW

```
task1 --> submit --> Receive task1 wait_for_finish() <&#8212; task1->state = READY <&#8212; Complete task1 task2 --> submit --> Receive task2 wait_for_finish() task2->state = READY <&#8212; Complete task2 task3 --> submit --> Receive task3 wait_for_finish() <&#8212; task3->state = READY <&#8212; Complete task3 task4 --> submit --> Receive task4 wait_for_finish() <&#8212; task4->state = READY <&#8212; Complete task4 NO MORE TASKS
```

Asynchronous: LVGL thread DRAW thread HW is IDLE task1 --> queue task1 submit --> Receive task1 task2 --> queue task2 is BUSY (with task1) task3 --> queue task3 still BUSY (with task1) task4 --> queue task4 becomes IDLE <&#8212; task1->state = READY <&#8212; Complete task1 submit --> Receive task2, task3, task4 NO MORE TASKS *wait\_for\_finish\_cb()* wait\_for\_finish() <&#8212; Complete task2, task3, task4 <&#8212; task2->state = READY <&#8212; <&#8212; task3->state = READY <&#8212; <&#8212; task4->state = READY <&#8212;

**Param draw\_unit**

**Return**

```
int32_t (*delete_cb)(lv_draw_unit_t *draw_unit)
```

Called to delete draw unit.

**Param draw\_unit**

**Return**

```
struct lv_draw_global_info_t
```

**Public Members**

```
lv_draw_unit_t *unit_head
```

```
uint32_t unit_cnt
```

```
uint32_t used_memory_for_layers
```

```
volatile int dispatch_req
```

```
lv_mutex_t circle_cache_mutex
```

```
bool task_running
```

**14.12.14 lv\_draw\_rect.h****Defines****LV\_RADIUS\_CIRCLE**

A very big radius to always draw as circle

## Functions

**void lv\_draw\_rect\_dsc\_init(*lv\_draw\_rect\_dsc\_t* \*dsc)**

Initialize a rectangle draw descriptor.

**Parametri**

**dsc** -- pointer to a draw descriptor

**void lv\_draw\_fill\_dsc\_init(*lv\_draw\_fill\_dsc\_t* \*dsc)**

Initialize a fill draw descriptor.

**Parametri**

**dsc** -- pointer to a draw descriptor

*lv\_draw\_fill\_dsc\_t* \***lv\_draw\_task\_get\_fill\_dsc(lv\_draw\_task\_t \*task)**

Try to get a fill draw descriptor from a draw task.

**Parametri**

**task** -- draw task

**Ritorna**

the task's draw descriptor or NULL if the task is not of type LV\_DRAW\_TASK\_TYPE\_FILL

**void lv\_draw\_fill(lv\_layer\_t \*layer, const *lv\_draw\_fill\_dsc\_t* \*dsc, const *lv\_area\_t* \*coords)**

Fill an area

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor variable
- **coords** -- the coordinates of the rectangle

**void lv\_draw\_border\_dsc\_init(*lv\_draw\_border\_dsc\_t* \*dsc)**

Initialize a border draw descriptor.

**Parametri**

**dsc** -- pointer to a draw descriptor

*lv\_draw\_border\_dsc\_t* \***lv\_draw\_task\_get\_border\_dsc(lv\_draw\_task\_t \*task)**

Try to get a border draw descriptor from a draw task.

**Parametri**

**task** -- draw task

**Ritorna**

the task's draw descriptor or NULL if the task is not of type LV\_DRAW\_TASK\_TYPE\_BORDER

**void lv\_draw\_border(lv\_layer\_t \*layer, const *lv\_draw\_border\_dsc\_t* \*dsc, const *lv\_area\_t* \*coords)**

Draw a border

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor variable
- **coords** -- the coordinates of the rectangle

**void lv\_draw\_box\_shadow\_dsc\_init(*lv\_draw\_box\_shadow\_dsc\_t* \*dsc)**

Initialize a box shadow draw descriptor.

**Parametri**

**dsc** -- pointer to a draw descriptor

*lv\_draw\_box\_shadow\_dsc\_t* \***lv\_draw\_task\_get\_box\_shadow\_dsc(lv\_draw\_task\_t \*task)**

Try to get a box shadow draw descriptor from a draw task.

**Parametri**

**task** -- draw task

**Ritorna**

the task's draw descriptor or NULL if the task is not of type LV\_DRAW\_TASK\_TYPE\_BOX\_SHADOW

```
void lv_draw_box_shadow(lv_layer_t *layer, const lv_draw_box_shadow_dsc_t *dsc, const lv_area_t *coords)
```

Draw a box shadow

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor variable
- **coords** -- the coordinates of the rectangle

```
void lv_draw_rect(lv_layer_t *layer, const lv_draw_rect_dsc_t *dsc, const lv_area_t *coords)
```

The rectangle is a wrapper for fill, border, bg. image and box shadow. Internally fill, border, image and box shadow draw tasks will be created.

**Parametri**

- **layer** -- pointer to a layer
- **dsc** -- pointer to an initialized draw descriptor variable
- **coords** -- the coordinates of the rectangle

```
struct lv_draw_rect_dsc_t
```

**Public Members**

*lv\_draw\_dsc\_base\_t* **base**

int32\_t **radius**

const void \***bg\_image\_src**

const void \***bg\_image\_symbol\_font**

*lv\_color\_t* **bg\_image\_recolor**

lv\_opa\_t **bg\_image\_opa**

lv\_opa\_t **bg\_image\_recolor\_opa**

uint8\_t **bg\_image\_tiled**

lv\_opa\_t **bg\_opa**

lv\_opa\_t **border\_opa**

lv\_opa\_t **outline\_opa**

lv\_opa\_t **shadow\_opa**

*lv\_color\_t* **bg\_color**

First element of a gradient is a color, so it maps well here

*lv\_grad\_dsc\_t* **bg\_grad**

---

```

lv_color_t border_color

int32_t border_width

lv_border_side_t border_side

uint8_t border_post

lv_color_t outline_color

int32_t outline_width

int32_t outline_pad

lv_color_t shadow_color

int32_t shadow_width

int32_t shadow_offset_x

int32_t shadow_offset_y

int32_t shadow_spread

struct lv_draw_fill_dsc_t

```

**Public Members***lv\_draw\_dsc\_base\_t* base

int32\_t radius

Radius, LV\_RADIUS\_CIRCLE for max. radius

lv\_opa\_t opa

Opacity in 0...255 range. LV\_OPA\_TRANSP, LV\_OPA\_10, LV\_OPA\_20, .. LV\_OPA\_COVER can be used as well

*lv\_color\_t* color

The color of the rectangle. If the gradient is set (grad.dir!=LV\_GRAD\_DIR\_NONE) it's ignored.

*lv\_grad\_dsc\_t* grad

Describe a gradient. If grad.dir is not LV\_GRAD\_DIR\_NONE color will be ignored

---

```

struct lv_draw_border_dsc_t

```

**Public Members*****lv\_draw\_dsc\_base\_t* base****int32\_t radius**

Radius, LV\_RADIUS\_CIRCLE for max. radius

***lv\_color\_t* color**

The color of the border.

**int32\_t width**

The width of the border in pixels

**lv\_opa\_t opa**

Opacity in 0...255 range. LV\_OPA\_TRANS, LV\_OPA\_10, LV\_OPA\_20, .. LV\_OPA\_COVER can be used as well

***lv\_border\_side\_t* side**

LV\_BORDER\_SIDE\_NONE/LEFT/RIGHT/TOP/BOTTOM/FULL. LV\_BORDER\_SIDE\_INTERNAL is an information for upper layers and shouldn't be used here.

struct **lv\_draw\_box\_shadow\_dsc\_t****Public Members*****lv\_draw\_dsc\_base\_t* base****int32\_t radius**

Radius, LV\_RADIUS\_CIRCLE for max. radius

***lv\_color\_t* color**

Color of the the shadow

**int32\_t width**

Width of the shadow. (radius of the blur)

**int32\_t spread**

Make the rectangle larger with this value in all directions. Can be negative too.

**int32\_t ofs\_x**

Offset the rectangle horizontally.

**int32\_t ofs\_y**

Offset the rectangle vertically.

**lv\_opa\_t opa**

Opacity in 0...255 range. LV\_OPA\_TRANS, LV\_OPA\_10, LV\_OPA\_20, .. LV\_OPA\_COVER can be used as well

**uint8\_t bg\_cover**

Set `bg_cover` to 1 if the background will cover the shadow. It's a hint to the renderer about it might skip some masking.

**14.12.15 lv\_draw\_rect\_private.h****14.12.16 lv\_draw\_triangle.h****Functions**

**void lv\_draw\_triangle\_dsc\_init(*lv\_draw\_triangle\_dsc\_t* \*draw\_dsc)**

Initialize a triangle draw descriptor

**Parametri**

**draw\_dsc** -- pointer to a draw descriptor

***lv\_draw\_triangle\_dsc\_t* \*lv\_draw\_task\_get\_triangle\_dsc(*lv\_draw\_task\_t* \*task)**

Try to get a triangle draw descriptor from a draw task.

**Parametri**

**task** -- draw task

**Ritorna**

the task's draw descriptor or NULL if the task is not of type LV\_DRAW\_TASK\_TYPE\_TRIANGLE

**void lv\_draw\_triangle(*lv\_layer\_t* \*layer, const *lv\_draw\_triangle\_dsc\_t* \*draw\_dsc)**

Create a triangle draw task

**Parametri**

- **layer** -- pointer to a layer
- **draw\_dsc** -- pointer to an initialized `lv_draw_triangle_dsc_t` object

**struct lv\_draw\_triangle\_dsc\_t**

**Public Members**

***lv\_draw\_dsc\_base\_t* base**

***lv\_point\_precise\_t* p[3]**

Points of the triangle. If LV\_USE\_FLOAT is enabled floats can be used here

***lv\_color\_t* color**

Color of the triangle

***lv\_opa\_t* opa**

Opacity of the arc in 0...255 range. LV\_OPA\_TRANSP, LV\_OPA\_10, LV\_OPA\_20, ..  
LV\_OPA\_COVER can be used as well

***lv\_grad\_dsc\_t* grad**

Describe a gradient. If `grad.dir` is not LV\_GRAD\_DIR\_NONE `color` will be ignored

**14.12.17 lv\_draw\_triangle\_private.h****14.12.18 lv\_draw\_vector.h****Typedef**

```
typedef void (*vector_draw_task_cb)(void *ctx, const lv_vector_path_t *path, const lv_vector_draw_dsc_t *dsc)
```

## Enums

enum **lv\_vector\_fill\_t**

*Values:*

enumerator **LV\_VECTOR\_FILL\_NONZERO**

enumerator **LV\_VECTOR\_FILL\_EVENODD**

enum **lv\_vector\_stroke\_cap\_t**

*Values:*

enumerator **LV\_VECTOR\_STROKE\_CAP\_BUTT**

enumerator **LV\_VECTOR\_STROKE\_CAP\_SQUARE**

enumerator **LV\_VECTOR\_STROKE\_CAP\_ROUND**

enum **lv\_vector\_stroke\_join\_t**

*Values:*

enumerator **LV\_VECTOR\_STROKE\_JOIN\_MITER**

enumerator **LV\_VECTOR\_STROKE\_JOIN\_BEVEL**

enumerator **LV\_VECTOR\_STROKE\_JOIN\_ROUND**

enum **lv\_vector\_path\_quality\_t**

*Values:*

enumerator **LV\_VECTOR\_PATH\_QUALITY\_MEDIUM**

enumerator **LV\_VECTOR\_PATH\_QUALITY\_HIGH**

enumerator **LV\_VECTOR\_PATH\_QUALITY\_LOW**

enum **lv\_vector\_blend\_t**

*Values:*

enumerator **LV\_VECTOR\_BLEND\_SRC\_OVER**

enumerator **LV\_VECTOR\_BLEND\_SRC\_IN**

enumerator **LV\_VECTOR\_BLEND\_DST\_OVER**

enumerator **LV\_VECTOR\_BLEND\_DST\_IN**

enumerator **LV\_VECTOR\_BLEND\_SCREEN**

enumerator **LV\_VECTOR\_BLEND\_MULTIPLY**

enumerator **LV\_VECTOR\_BLEND\_NONE**

enumerator **LV\_VECTOR\_BLEND\_ADDITIVE**

enumerator **LV\_VECTOR\_BLEND\_SUBTRACTIVE**

enum **lv\_vector\_path\_op\_t**

*Values:*

enumerator **LV\_VECTOR\_PATH\_OP\_MOVE\_TO**

enumerator **LV\_VECTOR\_PATH\_OP\_LINE\_TO**

enumerator **LV\_VECTOR\_PATH\_OP\_QUAD\_TO**

enumerator **LV\_VECTOR\_PATH\_OP\_CUBIC\_TO**

enumerator **LV\_VECTOR\_PATH\_OP\_CLOSE**

enum **lv\_vector\_draw\_style\_t**

*Values:*

enumerator **LV\_VECTOR\_DRAW\_STYLE\_SOLID**

enumerator **LV\_VECTOR\_DRAW\_STYLE\_PATTERN**

enumerator **LV\_VECTOR\_DRAW\_STYLE\_GRADIENT**

enum **lv\_vector\_gradient\_spread\_t**

*Values:*

enumerator **LV\_VECTOR\_GRADIENT\_SPREAD\_PAD**

enumerator **LV\_VECTOR\_GRADIENT\_SPREAD\_REPEAT**

enumerator **LV\_VECTOR\_GRADIENT\_SPREAD\_REFLECT**

enum **lv\_vector\_gradient\_style\_t**

*Values:*

enumerator **LV\_VECTOR\_GRADIENT\_STYLE\_LINEAR**

enumerator **LV\_VECTOR\_GRADIENT\_STYLE\_RADIAL**

enum **lv\_vector\_fill\_units\_t**

*Values:*

enumerator **LV\_VECTOR\_FILL\_UNITS\_OBJECT\_BOUNDING\_BOX**

enumerator **LV\_VECTOR\_FILL\_UNITS\_USER\_SPACE\_ON\_USE**

Relative coordinates relative to the object bounding box.

## Functions

void **lv\_matrix\_transform\_point**(const lv\_matrix\_t \*matrix, lv\_fpoint\_t \*point)

Transform the coordinates of a point using given matrix

**Parametri**

- **matrix** -- pointer to a matrix
- **point** -- pointer to a point

void **lv\_matrix\_transform\_path**(const lv\_matrix\_t \*matrix, lv\_vector\_path\_t \*path)

Transform all the coordinates of a path using given matrix

**Parametri**

- **matrix** -- pointer to a matrix
- **path** -- pointer to a path

lv\_vector\_path\_t \***lv\_vector\_path\_create**(*lv\_vector\_path\_quality\_t* quality)

Create a vector graphic path object

**Parametri**

- **quality** -- the quality hint of path

**Ritorna**

pointer to the created path object

void **lv\_vector\_path\_copy**(lv\_vector\_path\_t \*target\_path, const lv\_vector\_path\_t \*path)

Copy a path data to another

**Parametri**

- **target\_path** -- pointer to a path
- **path** -- pointer to source path

void **lv\_vector\_path\_clear**(lv\_vector\_path\_t \*path)

Clear path data

**Parametri**

- **path** -- pointer to a path

void **lv\_vector\_path\_delete**(lv\_vector\_path\_t \*path)

Delete the graphic path object

**Parametri**

- **path** -- pointer to a path

void **lv\_vector\_path\_move\_to**(lv\_vector\_path\_t \*path, const lv\_fpoint\_t \*p)

Begin a new sub path and set a point to path

**Parametri**

- **path** -- pointer to a path
- **p** -- pointer to a **lv\_fpoint\_t** variable

**void lv\_vector\_path\_line\_to(lv\_vector\_path\_t \*path, const lv\_fpoint\_t \*p)**

Add a line to the path from last point to the point

**Parametri**

- **path** -- pointer to a path
- **p** -- pointer to a `lv_fpoint_t` variable

**void lv\_vector\_path\_quad\_to(lv\_vector\_path\_t \*path, const lv\_fpoint\_t \*p1, const lv\_fpoint\_t \*p2)**

Add a quadratic bezier line to the path from last point to the point

**Parametri**

- **path** -- pointer to a path
- **p1** -- pointer to a `lv_fpoint_t` variable for control point
- **p2** -- pointer to a `lv_fpoint_t` variable for end point

**void lv\_vector\_path\_cubic\_to(lv\_vector\_path\_t \*path, const lv\_fpoint\_t \*p1, const lv\_fpoint\_t \*p2, const lv\_fpoint\_t \*p3)**

Add a cubic bezier line to the path from last point to the point

**Parametri**

- **path** -- pointer to a path
- **p1** -- pointer to a `lv_fpoint_t` variable for first control point
- **p2** -- pointer to a `lv_fpoint_t` variable for second control point
- **p3** -- pointer to a `lv_fpoint_t` variable for end point

**void lv\_vector\_path\_arc\_to(lv\_vector\_path\_t \*path, float radius\_x, float radius\_y, float rotate\_angle, bool large\_arc, bool clockwise, const lv\_fpoint\_t \*p)**

Add ellipse arc to the path from last point to the point

**Parametri**

- **path** -- pointer to a path
- **radius\_x** -- the x radius for ellipse arc
- **radius\_y** -- the y radius for ellipse arc
- **rotate\_angle** -- the rotate angle for arc
- **large\_arc** -- true for large arc, otherwise small
- **clockwise** -- true for clockwise, otherwise anticlockwise
- **p** -- pointer to a `lv_fpoint_t` variable for end point

**void lv\_vector\_path\_close(lv\_vector\_path\_t \*path)**

Close the sub path

**Parametri**

**path** -- pointer to a path

**void lv\_vector\_path\_get\_bounding(const lv\_vector\_path\_t \*path, lv\_area\_t \*area)**

Get the bounding box of a path

**Parametri**

- **path** -- pointer to a path
- **area** -- pointer to a `lv_area_t` variable for bounding box

**void lv\_vector\_path\_append\_rect(lv\_vector\_path\_t \*path, const lv\_area\_t \*rect, float rx, float ry)**

Add a rectangle to the path

**Parametri**

- **path** -- pointer to a path
- **rect** -- pointer to a `lv_area_t` variable
- **rx** -- the horizontal radius for rounded rectangle
- **ry** -- the vertical radius for rounded rectangle

**void lv\_vector\_path\_append\_circle(lv\_vector\_path\_t \*path, const lv\_fpoint\_t \*c, float rx, float ry)**

Add a circle to the path

**Parametri**

- **path** -- pointer to a path
- **c** -- pointer to a `lv_fpoint_t` variable for center of the circle
- **rx** -- the horizontal radius for circle
- **ry** -- the vertical radius for circle

---

```
void lv_vector_path_append_arc(lv_vector_path_t *path, const lv_fpoint_t *c, float radius, float
                               start_angle, float sweep, bool pie)
```

Add a arc to the path

**Parametri**

- **path** -- pointer to a path
- **c** -- pointer to a `lv_fpoint_t` variable for center of the circle
- **radius** -- the radius for arc
- **start\_angle** -- the start angle for arc
- **sweep** -- the sweep angle for arc, could be negative
- **pie** -- true: draw a pie, false: draw a arc

```
void lv_vector_path_append_path(lv_vector_path_t *path, const lv_vector_path_t *subpath)
```

Add an sub path to the path

**Parametri**

- **path** -- pointer to a path
- **subpath** -- pointer to another path which will be added

```
lv_vector_dsc_t *lv_vector_dsc_create(lv_layer_t *layer)
```

Create a vector graphic descriptor

**Parametri**

**layer** -- pointer to a layer

**Ritorna**

pointer to the created descriptor

```
void lv_vector_dsc_delete(lv_vector_dsc_t *dsc)
```

Delete the vector graphic descriptor

**Parametri**

**dsc** -- pointer to a vector graphic descriptor

```
void lv_vector_dsc_set_transform(lv_vector_dsc_t *dsc, const lv_matrix_t *matrix)
```

Set a matrix to current transformation matrix

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **matrix** -- pointer to a matrix

```
void lv_vector_dsc_set_blend_mode(lv_vector_dsc_t *dsc, lv_vector_blend_t blend)
```

Set blend mode for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **blend** -- the blend mode to be set in `lv_vector_blend_t`

```
void lv_vector_dsc_set_fill_color32(lv_vector_dsc_t *dsc, lv_color32_t color)
```

Set fill color for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **color** -- the color to be set in `lv_color32_t` format

```
void lv_vector_dsc_set_fill_color(lv_vector_dsc_t *dsc, lv_color_t color)
```

Set fill color for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **color** -- the color to be set in `lv_color_t` format

```
void lv_vector_dsc_set_fill_opa(lv_vector_dsc_t *dsc, lv_opa_t opa)
```

Set fill opacity for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **opa** -- the opacity to be set in `lv_opa_t` format

**void lv\_vector\_dsc\_set\_fill\_rule(lv\_vector\_dsc\_t \*dsc, *lv\_vector\_fill\_t* rule)**

Set fill rule for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **rule** -- the fill rule to be set in *lv\_vector\_fill\_t* format

**void lv\_vector\_dsc\_set\_fill\_units(lv\_vector\_dsc\_t \*dsc, const *lv\_vector\_fill\_units\_t* units)**

Set the fill units for descriptor.

**i Nota**

The units can be either relative to the object bounding box or absolute in user space. This API specifically affects the drawing position of the fill image and does not impact other elements.

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **units** -- the units to be set in *lv\_vector\_fill\_units\_t* format

**void lv\_vector\_dsc\_set\_fill\_image(lv\_vector\_dsc\_t \*dsc, const lv\_draw\_image\_dsc\_t \*img\_dsc)**

Set fill image for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **img\_dsc** -- pointer to a *lv\_draw\_image\_dsc\_t* variable

**void lv\_vector\_dsc\_set\_fill\_linear\_gradient(lv\_vector\_dsc\_t \*dsc, float x1, float y1, float x2, float y2)**

Set fill linear gradient for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **x1** -- the x for start point
- **y1** -- the y for start point
- **x2** -- the x for end point
- **y2** -- the y for end point

**void lv\_vector\_dsc\_set\_fill\_radial\_gradient(lv\_vector\_dsc\_t \*dsc, float cx, float cy, float radius)**

Set fill radial gradient radius for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **cx** -- the x for center of the circle
- **cy** -- the y for center of the circle
- **radius** -- the radius for circle

**void lv\_vector\_dsc\_set\_fill\_gradient\_spread(lv\_vector\_dsc\_t \*dsc, *lv\_vector\_gradient\_spread\_t* spread)**

Set fill radial gradient spread for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **spread** -- the gradient spread to be set in *lv\_vector\_gradient\_spread\_t* format

**void lv\_vector\_dsc\_set\_fill\_gradient\_color\_stops(lv\_vector\_dsc\_t \*dsc, const *lv\_grad\_stop\_t* \*stops, uint16\_t count)**

Set fill gradient color stops for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **stops** -- an array of *lv\_grad\_stop\_t* variables
- **count** -- the number of stops in the array, range: 0..LV\_GRADIENT\_MAX\_STOPS

**void lv\_vector\_dsc\_set\_fill\_transform(lv\_vector\_dsc\_t \*dsc, const lv\_matrix\_t \*matrix)**

Set a matrix to current fill transformation matrix

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **matrix** -- pointer to a matrix

**void lv\_vector\_dsc\_set\_stroke\_color32(lv\_vector\_dsc\_t \*dsc, lv\_color32\_t color)**

Set stroke color for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **color** -- the color to be set in *lv\_color32\_t* format

**void lv\_vector\_dsc\_set\_stroke\_color(lv\_vector\_dsc\_t \*dsc, lv\_color\_t color)**

Set stroke color for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **color** -- the color to be set in *lv\_color\_t* format

**void lv\_vector\_dsc\_set\_stroke\_opa(lv\_vector\_dsc\_t \*dsc, lv\_opa\_t opa)**

Set stroke opacity for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **opa** -- the opacity to be set in *lv\_opa\_t* format

**void lv\_vector\_dsc\_set\_stroke\_width(lv\_vector\_dsc\_t \*dsc, float width)**

Set stroke line width for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **width** -- the stroke line width

**void lv\_vector\_dsc\_set\_stroke\_dash(lv\_vector\_dsc\_t \*dsc, float \*dash\_pattern, uint16\_t dash\_count)**

Set stroke line dash pattern for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **dash\_pattern** -- an array of values that specify the segments of dash line
- **dash\_count** -- the length of dash pattern array

**void lv\_vector\_dsc\_set\_stroke\_cap(lv\_vector\_dsc\_t \*dsc, lv\_vector\_stroke\_cap\_t cap)**

Set stroke line cap style for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **cap** -- the line cap to be set in *lv\_vector\_stroke\_cap\_t* format

**void lv\_vector\_dsc\_set\_stroke\_join(lv\_vector\_dsc\_t \*dsc, lv\_vector\_stroke\_join\_t join)**

Set stroke line join style for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **join** -- the line join to be set in *lv\_vector\_stroke\_join\_t* format

**void lv\_vector\_dsc\_set\_stroke\_miter\_limit(lv\_vector\_dsc\_t \*dsc, uint16\_t miter\_limit)**

Set stroke miter limit for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **miter\_limit** -- the stroke miter\_limit

**void lv\_vector\_dsc\_set\_stroke\_linear\_gradient(lv\_vector\_dsc\_t \*dsc, float x1, float y1, float x2, float y2)**

Set stroke linear gradient for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor

- **x1** -- the x for start point
- **y1** -- the y for start point
- **x2** -- the x for end point
- **y2** -- the y for end point

**void lv\_vector\_dsc\_set\_stroke\_radial\_gradient(lv\_vector\_dsc\_t \*dsc, float cx, float cy, float radius)**

Set stroke radial gradient for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **cx** -- the x for center of the circle
- **cy** -- the y for center of the circle
- **radius** -- the radius for circle

**void lv\_vector\_dsc\_set\_stroke\_gradient\_spread(lv\_vector\_dsc\_t \*dsc, lv\_vector\_gradient\_spread\_t spread)**

Set stroke color stops for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **spread** -- the gradient spread to be set in lv\_vector\_gradient\_spread\_t format

**void lv\_vector\_dsc\_set\_stroke\_gradient\_color\_stops(lv\_vector\_dsc\_t \*dsc, const lv\_grad\_stop\_t \*stops, uint16\_t count)**

Set stroke color stops for descriptor

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **stops** -- an array of *lv\_grad\_stop\_t* variables
- **count** -- the number of stops in the array

**void lv\_vector\_dsc\_set\_stroke\_transform(lv\_vector\_dsc\_t \*dsc, const lv\_matrix\_t \*matrix)**

Set a matrix to current stroke transformation matrix

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **matrix** -- pointer to a matrix

**void lv\_vector\_dsc\_identity(lv\_vector\_dsc\_t \*dsc)**

Set current transformation matrix to identity matrix

**Parametri**

**dsc** -- pointer to a vector graphic descriptor

**void lv\_vector\_dsc\_scale(lv\_vector\_dsc\_t \*dsc, float scale\_x, float scale\_y)**

Change the scale factor of current transformation matrix

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **scale\_x** -- the scale factor for the X direction
- **scale\_y** -- the scale factor for the Y direction

**void lv\_vector\_dsc\_rotate(lv\_vector\_dsc\_t \*dsc, float degree)**

Rotate current transformation matrix with origin

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **degree** -- angle to rotate

**void lv\_vector\_dsc\_translate(lv\_vector\_dsc\_t \*dsc, float tx, float ty)**

Translate current transformation matrix to new position

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **tx** -- the amount of translate in x direction
- **ty** -- the amount of translate in y direction

**void lv\_vector\_dsc\_skew(lv\_vector\_dsc\_t \*dsc, float skew\_x, float skew\_y)**

Change the skew factor of current transformation matrix

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **skew\_x** -- the skew factor for x direction
- **skew\_y** -- the skew factor for y direction

**void lv\_vector\_dsc\_add\_path(lv\_vector\_dsc\_t \*dsc, const lv\_vector\_path\_t \*path)**

Add a graphic path to the draw list

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **path** -- pointer to a path

**void lv\_vector\_clear\_area(lv\_vector\_dsc\_t \*dsc, const lv\_area\_t \*rect)**

Clear a rectangle area use current fill color

**Parametri**

- **dsc** -- pointer to a vector graphic descriptor
- **rect** -- the area to clear in the buffer

**void lv\_draw\_vector(lv\_vector\_dsc\_t \*dsc)**

Draw all the vector graphic paths

**Parametri**

**dsc** -- pointer to a vector graphic descriptor

struct **\_lv\_fpoint\_t**

**Public Members**

float **x**

float **y**

### 14.12.19 lv\_draw\_vector\_private.h

**Functions**

**void lv\_vector\_for\_each\_destroy\_tasks(lv\_ll\_t \*task\_list, vector\_draw\_task\_cb cb, void \*data)**

struct **\_lv\_vector\_path\_t**

**Public Members**

**lv\_vector\_path\_quality\_t quality**

**lv\_array\_t ops**

**lv\_array\_t points**

struct **\_lv\_vector\_gradient\_t**

## Public Members

*lv\_vector\_gradient\_style\_t* **style**

*lv\_grad\_stop\_t* **stops[2]**

A gradient stop array

**uint16\_t stops\_count**

The number of used stops in the array

float **x1**

float **y1**

float **x2**

float **y2**

float **cx**

float **cy**

float **cr**

*lv\_vector\_gradient\_spread\_t* **spread**

struct **\_lv\_vector\_fill\_dsc\_t**

## Public Members

*lv\_vector\_draw\_style\_t* **style**

*lv\_color32\_t* **color**

*lv\_opa\_t* **opa**

*lv\_vector\_fill\_t* **fill\_rule**

*lv\_vector\_fill\_units\_t* **fill\_units**

*lv\_draw\_image\_dsc\_t* **img\_dsc**

*lv\_vector\_gradient\_t* **gradient**

*lv\_matrix\_t* **matrix**

struct **\_lv\_vector\_stroke\_dsc\_t**

**Public Members**`lv_vector_draw_style_t style``lv_color32_t color``lv_opa_t opa``float width``lv_array_t dash_pattern``lv_vector_stroke_cap_t cap``lv_vector_stroke_join_t join``uint16_t miter_limit``lv_vector_gradient_t gradient``lv_matrix_t matrix``struct _lv_vector_draw_dsc_t`**Public Members**`lv_vector_fill_dsc_t fill_dsc``lv_vector_stroke_dsc_t stroke_dsc``lv_matrix_t matrix``lv_vector_blend_t blend_mode``lv_area_t scissor_area``struct _lv_draw_vector_task_dsc_t`**Public Members**`lv_draw_dsc_base_t base``lv_ll_t *task_list``struct _lv_vector_dsc_t`

**Public Members**

```
lv_layer_t *layer
lv_vector_draw_dsc_t current_dsc
lv_draw_vector_task_dsc_t tasks
```

**14.12.20 lv\_image\_decoder.h****TypeDef**

**typedef lv\_result\_t (\*lv\_image\_decoder\_info\_f\_t)(lv\_image\_decoder\_t \*decoder,  
lv\_image\_decoder\_dsc\_t \*dsc, lv\_image\_header\_t \*header)**

Get info from an image and store in the header

**Param decoder**

pointer to decoder object

**Param dsc**

pointer to decoder descriptor

**Param header**

store the info here

**Return**

LV\_RESULT\_OK: info written correctly; LV\_RESULT\_INVALID: failed

**typedef lv\_result\_t (\*lv\_image\_decoder\_open\_f\_t)(lv\_image\_decoder\_t \*decoder,  
lv\_image\_decoder\_dsc\_t \*dsc)**

Open an image for decoding. Prepare it as it is required to read it later

**Param decoder**

pointer to the decoder the function associated with

**Param dsc**

pointer to decoder descriptor. `src`, `color` are already initialized in it.

**typedef lv\_result\_t (\*lv\_image\_decoder\_get\_area\_cb\_t)(lv\_image\_decoder\_t \*decoder,  
lv\_image\_decoder\_dsc\_t \*dsc, const lv\_area\_t \*full\_area, lv\_area\_t \*decoded\_area)**

Decode `full_area` pixels incrementally by calling in a loop. Set `decoded_area` values to `LV_COORD_MIN` on first call. Required only if the "open" function can't return with the whole decoded pixel array.

**Param decoder**

pointer to the decoder the function associated with

**Param dsc**

pointer to decoder descriptor

**Param full\_area**

input parameter. the full area to decode after enough subsequent calls

**Param decoded\_area**

input+output parameter. set the values to `LV_COORD_MIN` for the first call and to reset decoding. the decoded area is stored here after each call.

**Return**

LV\_RESULT\_OK: ok; LV\_RESULT\_INVALID: failed or there is nothing left to decode

**typedef void (\*lv\_image\_decoder\_close\_f\_t)(lv\_image\_decoder\_t \*decoder, lv\_image\_decoder\_dsc\_t  
\*dsc)**

Close the pending decoding. Free resources etc.

**Param decoder**

pointer to the decoder the function associated with

**Param dsc**

pointer to decoder descriptor

```
typedef void (*lv_image_decoder_custom_draw_t)(lv_layer_t *layer, const lv_image_decoder_dsc_t
*dsc, const lv_area_t *coords, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *clip_area)
```

Custom drawing functions for special image formats.

**Param layer**

pointer to a layer

**Param dsc**

pointer to decoder descriptor

**Param coords**

the coordinates of the image

**Param draw\_dsc**

the draw image descriptor

**Param clip\_area**

the clip area of the image

**Enums****enum `lv_image_src_t`**

Source of image.

*Values:*

enumerator **LV\_IMAGE\_SRC\_VARIABLE**

enumerator **LV\_IMAGE\_SRC\_FILE**

Binary/C variable

enumerator **LV\_IMAGE\_SRC\_SYMBOL**

File in filesystem

enumerator **LV\_IMAGE\_SRC\_UNKNOWN**

Symbol (lv\_symbol\_def.h)

**Functions****lv\_result\_t `lv_image_decoder_get_info`(const void \*src, `lv_image_header_t` \*header)**

Get information about an image. Try the created image decoder one by one. Once one is able to get info that info will be used.

**Parametri**

- **src** -- the image source. Can be 1) File name: E.g. "S:folder/img1.png" (The drivers needs to registered via `lv_fs_drv_register()`) 2) Variable: Pointer to an `lv_image_dsc_t` variable 3) Symbol: E.g. LV\_SYMBOL\_OK
- **header** -- the image info will be stored here

**Ritorna**

LV\_RESULT\_OK: success; LV\_RESULT\_INVALID: wasn't able to get info about the image

**lv\_result\_t `lv_image_decoder_open`(lv\_image\_decoder\_dsc\_t \*dsc, const void \*src, const `lv_image_decoder_args_t` \*args)**

Open an image. Try the created image decoders one by one. Once one is able to open the image that decoder is saved in `dsc`

**Parametri**

- **dsc** -- describes a decoding session. Simply a pointer to an `lv_image_decoder_dsc_t` variable.

- **src** -- the image source. Can be 1) File name: E.g. "S:folder/img1.png" (The drivers needs to registered via `lv_fs_drv_register()`) 2) Variable: Pointer to an `lv_image_dsc_t` variable 3) Symbol: E.g. LV\_SYMBOL\_OK
- **args** -- args about how the image should be opened.

**Ritorna**

`LV_RESULT_OK`: opened the image. `dsc->decoded` and `dsc->header` are set.  
`LV_RESULT_INVALID`: none of the registered image decoders were able to open the image.

```
lv_result_t lv_image_decoder_get_area(lv_image_decoder_dsc_t *dsc, const lv_area_t *full_area,
                                      lv_area_t *decoded_area)
```

Decode `full_area` pixels incrementally by calling in a loop. Set `decoded_area` to `LV_COORD_MIN` on first call.

**Parametri**

- **dsc** -- image decoder descriptor
- **full\_area** -- input parameter. the full area to decode after enough subsequent calls
- **decoded\_area** -- input+output parameter. set the values to `LV_COORD_MIN` for the first call and to reset decoding. the decoded area is stored here after each call.

**Ritorna**

`LV_RESULT_OK`: success; `LV_RESULT_INVALID`: an error occurred or there is nothing left to decode

```
void lv_image_decoder_close(lv_image_decoder_dsc_t *dsc)
```

Close a decoding session

**Parametri**

**dsc** -- pointer to `lv_image_decoder_dsc_t` used in `lv_image_decoder_open`

```
lv_image_decoder_t *lv_image_decoder_create(void)
```

Create a new image decoder

**Ritorna**

pointer to the new image decoder

```
void lv_image_decoder_delete(lv_image_decoder_t *decoder)
```

Delete an image decoder

**Parametri**

**decoder** -- pointer to an image decoder

```
lv_image_decoder_t *lv_image_decoder_get_next(lv_image_decoder_t *decoder)
```

Get the next image decoder in the linked list of image decoders

**Parametri**

**decoder** -- pointer to an image decoder or NULL to get the first one

**Ritorna**

the next image decoder or NULL if no more image decoder exists

```
void lv_image_decoder_set_info_cb(lv_image_decoder_t *decoder, lv_image_decoder_info_f_t
                                  info_cb)
```

Set a callback to get information about the image

**Parametri**

- **decoder** -- pointer to an image decoder
- **info\_cb** -- a function to collect info about an image (fill an `lv_image_header_t` struct)

```
void lv_image_decoder_set_open_cb(lv_image_decoder_t *decoder, lv_image_decoder_open_f_t
                                  open_cb)
```

Set a callback to open an image

**Parametri**

- **decoder** -- pointer to an image decoder
- **open\_cb** -- a function to open an image

```
void lv_image_decoder_set_get_area_cb(lv_image_decoder_t *decoder,
                                      lv_image_decoder_get_area_cb_t read_line_cb)
```

Set a callback to a decoded line of an image

**Parametri**

- **decoder** -- pointer to an image decoder
- **read\_line\_cb** -- a function to read a line of an image

```
void lv_image_decoder_set_close_cb(lv_image_decoder_t *decoder, lv_image_decoder_close_f_t
                                   close_cb)
```

Set a callback to close a decoding session. E.g. close files and free other resources.

**Parametri**

- **decoder** -- pointer to an image decoder
- **close\_cb** -- a function to close a decoding session

```
lv_cache_entry_t *lv_image_decoder_add_to_cache(lv_image_decoder_t *decoder,
                                                lv_image_cache_data_t *search_key, const
                                                lv_draw_buf_t *decoded, void *user_data)
```

```
lv_draw_buf_t *lv_image_decoder_post_process(lv_image_decoder_dsc_t *dsc, lv_draw_buf_t
                                             *decoded)
```

Check the decoded image, make any modification if decoder **args** requires.

 **Nota**

A new draw buf will be allocated if provided **decoded** is not modifiable or stride mismatch etc.

**Parametri**

- **dsc** -- pointer to a decoder descriptor
- **decoded** -- pointer to a decoded image to post process to meet dsc->args requirement.

**Ritorna**

post processed draw buffer, when it differs with **decoded**, it's newly allocated.

## 14.12.21 lv\_image\_decoder\_private.h

### Functions

```
void lv_image_decoder_init(uint32_t image_cache_size, uint32_t image_header_count)
```

Initialize the image decoder module

**Parametri**

- **image\_cache\_size** -- Image cache size in bytes. 0 to disable cache.
- **image\_header\_count** -- Number of header cache entries. 0 to disable header cache.

```
void lv_image_decoder_deinit(void)
```

Deinitialize the image decoder module

```
struct _lv_image_decoder_args_t
```

#include <lv\_image\_decoder\_private.h> Image decoder args. It determines how to decoder an image, e.g. whether to premultiply the alpha or not. It should be passed to lv\_img\_decoder\_open() function. If NULL is provided, default args are used.

Default args: all field are zero or false.

**Public Members****bool stride\_align**

Whether stride should be aligned

**bool premultiply**

Whether image should be premultiplied or not after decoding

**bool no\_cache**

When set, decoded image won't be put to cache, and decoder open will also ignore cache.

**bool use\_indexed**

Decoded indexed image as is. Convert to ARGB8888 if false.

**bool flush\_cache**

Whether to flush the data cache after decoding

**struct \_lv\_image\_decoder\_t****Public Members***lv\_image\_decoder\_info\_f\_t* **info\_cb***lv\_image\_decoder\_open\_f\_t* **open\_cb***lv\_image\_decoder\_get\_area\_cb\_t* **get\_area\_cb***lv\_image\_decoder\_close\_f\_t* **close\_cb***lv\_image\_decoder\_custom\_draw\_t* **custom\_draw\_cb**const char \***name**void \***user\_data****struct \_lv\_image\_cache\_data\_t****Public Members***lv\_cache\_slot\_size\_t* **slot**const void \***src***lv\_image\_src\_t* **src\_type**const lv\_draw\_buf\_t \***decoded**

```
const lv_image_decoder_t *decoder
void *user_data
struct _lv_image_header_cache_data_t
```

**Public Members**

```
const void *src
lv_image_src_t src_type
lv_image_header_t header
lv_image_decoder_t *decoder
struct _lv_image_decoder_dsc_t
#include <lv_image_decoder_private.h> Describe an image decoding session. Stores data about the decoding
```

**Public Members****lv\_image\_decoder\_t \*decoder**

The decoder which was able to open the image source

**lv\_image\_decoder\_args\_t args**

A copy of parameters of how this image is decoded

**const void \*src**

The image source. A file path like "S:my\_img.png" or pointer to an *lv\_image\_dsc\_t* variable

**lv\_image\_src\_t src\_type**

Type of the source: file or variable. Can be set in **open** function if required

**lv\_fs\_file\_t file****lv\_image\_header\_t header**

Info about the opened image: color format, size, etc. MUST be set in **open** function

**const lv\_draw\_buf\_t \*decoded**

Pointer to a draw buffer where the image's data (pixels) are stored in a decoded, plain format. MUST be set in **open** or **get\_area\_cb** function

**const lv\_color32\_t \*palette****uint32\_t palette\_size**

`uint32_t time_to_open`

How much time did it take to open the image. [ms] If not set `lv_image_cache` will measure and set the time to open

`const char *error_msg`

A text to display instead of the image when the image can't be opened. Can be set in `open` function or set NULL.

`lv_cache_t *cache`

`lv_cache_entry_t *cache_entry`

Point to cache entry information

`void *user_data`

Store any custom data here is required

## 14.12.22 lv\_image\_dsc.h

### Defines

`LV_IMAGE_HEADER_MAGIC`

Magic number for lvgl image, 9 means lvgl version 9 It must be neither a valid ASCII character nor larger than 0x80. See `lv_image_src_get_type`.

`LV_IMAGE_FLAGS_USER_MASK`

Flags reserved for user, lvgl won't use these bits.

### Typedef

`typedef enum _lvimage_flags_t lv_image_flags_t`

### Enums

`enum _lvimage_flags_t`

*Values:*

enumerator `LV_IMAGE_FLAGS_PREMULTIPLIED`

For RGB map of the image data, mark if it's pre-multiplied with alpha. For indexed image, this bit indicated palette data is pre-multiplied with alpha.

enumerator `LV_IMAGE_FLAGS_COMPRESSED`

The image data is compressed, so decoder needs to decode image firstly. If this flag is set, the whole image will be decompressed upon decode, and `get_area_cb` won't be necessary.

enumerator `LV_IMAGE_FLAGS_ALLOCATED`

The image is allocated from heap, thus should be freed after use.

enumerator `LV_IMAGE_FLAGS_MODIFIABLE`

If the image data is malloced and can be processed in place. In image decoder post processing, this flag means we modify it in-place.

enumerator **LV\_IMAGE\_FLAGS\_CUSTOM\_DRAW**

The image has custom drawing methods.

enumerator **LV\_IMAGE\_FLAGS\_USER1**

Flags reserved for user, lvgl won't use these bits.

enumerator **LV\_IMAGE\_FLAGS\_USER2**

enumerator **LV\_IMAGE\_FLAGS\_USER3**

enumerator **LV\_IMAGE\_FLAGS\_USER4**

enumerator **LV\_IMAGE\_FLAGS\_USER5**

enumerator **LV\_IMAGE\_FLAGS\_USER6**

enumerator **LV\_IMAGE\_FLAGS\_USER7**

enumerator **LV\_IMAGE\_FLAGS\_USER8**

enum **lv\_image\_compress\_t**

*Values:*

enumerator **LV\_IMAGE\_COMPRESS\_NONE**

enumerator **LV\_IMAGE\_COMPRESS\_RLE**

LVGL custom RLE compression

enumerator **LV\_IMAGE\_COMPRESS\_LZ4**

struct **lv\_image\_header\_t**

### Public Members

`uint32_t magic`

Magic number. Must be LV\_IMAGE\_HEADER\_MAGIC

`uint32_t cf`

Color format: See `lv_color_format_t`

`uint32_t flags`

Image flags, see `lv_image_flags_t`

`uint32_t w`

`uint32_t h`

`uint32_t stride`  
Number of bytes in a row

`uint32_t reserved_2`  
Reserved to be used later

struct `lv_yuv_plane_t`

#### Public Members

`void *buf`

`uint32_t stride`  
Number of bytes in a row

union `lv_yuv_buf_t`

#### Public Members

`lv_yuv_plane_t yuv`

packed format

`lv_yuv_plane_t y`

`lv_yuv_plane_t u`

`lv_yuv_plane_t v`

struct `lv_yuv_buf_t planar`  
planar format with 3 plane

`lv_yuv_plane_t uv`

struct `lv_yuv_buf_t semi_planar`  
planar format with 2 plane

struct `lv_image_dsc_t`

#include <lv\_image\_dsc.h> Struct to describe a constant image resource. It's similar to lv\_draw\_buf\_t, but the data is constant.

#### Public Members

`lv_image_header_t header`

A header describing the basics of the image

`uint32_t data_size`

Size of the image in bytes

```
const uint8_t *data
    Pointer to the data of the image

const void *reserved
    A reserved field to make it has same size as lv_draw_buf_t

const void *reserved_2
    A reserved field to make it has same size as lv_draw_buf_t
```

### 14.12.23 dma2d

**lv\_draw\_dma2d.h**

#### Functions

```
void lv_draw_dma2d_init(void)
void lv_draw_dma2d_deinit(void)
void lv_draw_dma2d_transfer_complete_interrupt_handler(void)
```

**lv\_draw\_dma2d\_private.h**

#### Defines

**LV\_DRAW\_DMA2D\_ASYNC**

**LV\_DRAW\_DMA2D\_CACHE**

#### Enums

enum **lv\_draw\_dma2d\_output\_cf\_t**

*Values:*

enumerator **LV\_DRAW\_DMA2D\_OUTPUT\_CF\_ARGB8888**

enumerator **LV\_DRAW\_DMA2D\_OUTPUT\_CF\_RGB888**

enumerator **LV\_DRAW\_DMA2D\_OUTPUT\_CF\_RGB565**

enumerator **LV\_DRAW\_DMA2D\_OUTPUT\_CF\_ARGB1555**

enumerator **LV\_DRAW\_DMA2D\_OUTPUT\_CF\_ARGB4444**

enum **lv\_draw\_dma2d\_fgbg\_cf\_t**

*Values:*

enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_ARGB8888**

enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_RGB888**

enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_RGB565**  
enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_ARGB1555**  
enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_ARGB4444**  
enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_L8**  
enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_AL44**  
enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_AL88**  
enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_L4**  
enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_A8**  
enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_A4**  
enumerator **LV\_DRAW\_DMA2D\_FGBG\_CF\_YCBCR**

enum **lv\_draw\_dma2d\_mode\_t**

*Values:*

enumerator **LV\_DRAW\_DMA2D\_MODE\_MEMORY\_TO\_MEMORY**  
enumerator **LV\_DRAW\_DMA2D\_MODE\_MEMORY\_TO\_MEMORY\_WITH\_PFC**  
enumerator **LV\_DRAW\_DMA2D\_MODE\_MEMORY\_TO\_MEMORY\_WITH\_BLENDING**  
enumerator **LV\_DRAW\_DMA2D\_MODE\_REGISTER\_TO\_MEMORY**  
enumerator  
**LV\_DRAW\_DMA2D\_MODE\_MEMORY\_TO\_MEMORY\_WITH\_BLENDING\_AND\_FIXED\_COLOR\_FG**  
enumerator  
**LV\_DRAW\_DMA2D\_MODE\_MEMORY\_TO\_MEMORY\_WITH\_BLENDING\_AND\_FIXED\_COLOR\_BG**

enum **lv\_draw\_dma2d\_alpha\_mode\_t**

*Values:*

enumerator **LV\_DRAW\_DMA2D\_ALPHA\_MODE\_NO MODIFY IMAGE\_ALPHA\_CHANNEL**  
enumerator **LV\_DRAW\_DMA2D\_ALPHA\_MODE\_REPLACE\_ALPHA\_CHANNEL**  
enumerator **LV\_DRAW\_DMA2D\_ALPHA\_MODE\_MULTIPLY\_IMAGE\_ALPHA\_CHANNEL**

## Functions

```
void lv_draw_dma2d_opaque_fill(lv_draw_task_t *t, void *first_pixel, int32_t w, int32_t h, int32_t stride)

void lv_draw_dma2d_fill(lv_draw_task_t *t, void *first_pixel, int32_t w, int32_t h, int32_t stride)

void lv_draw_dma2d_opaque_image(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)

void lv_draw_dma2d_image(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)

lv_draw_dma2d_output_cf_t lv_draw_dma2d_cf_to_dma2d_output_cf(lv_color_format_t cf)

uint32_t lv_draw_dma2d_color_to_dma2d_color(lv_draw_dma2d_output_cf_t cf, lv_color_t color)

void lv_draw_dma2d_configure_and_start_transfer(const lv_draw_dma2d_configuration_t *conf)
```

struct **lv\_draw\_dma2d\_configuration\_t**

### Public Members

```
lv_draw_dma2d_mode_t mode

uint32_t w

uint32_t h

void *output_address

uint32_t output_offset

lv_draw_dma2d_output_cf_t output_cf

uint32_t reg_to_mem_mode_color

const void *fg_address

uint32_t fg_offset

lv_draw_dma2d_fgbg_cf_t fg_cf

uint32_t fg_color

uint32_t fg_alpha_mode

uint32_t fg_alpha

const void *bg_address
```

```

    uint32_t bg_offset

    lv_draw_dma2d_fgbg_cf_t bg_cf

    uint32_t bg_color

    uint32_t bg_alpha_mode

    uint32_t bg_alpha

struct lv_draw_dma2d_cache_area_t

```

**Public Members**

```

    const void *first_byte

    uint32_t width_bytes

    uint32_t height

    uint32_t stride

```

```
struct lv_draw_dma2d_unit_t
```

**Public Members**

```

    lv_draw_unit_t base_unit

    lv_draw_task_t *volatile task_act

```

**14.12.24 espressif**

**ppa**  
*lv\_draw\_ppa.h*

**Functions**

```

void lv_draw_ppa_init(void)

void lv_draw_ppa_deinit(void)

void lv_draw_buf_ppa_init_handlers(void)

void lv_draw_ppa_fill(lv_draw_task_t *t, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)

void lv_draw_ppa_img(lv_draw_task_t *t, const lv_draw_image_dsc_t *dsc, const lv_area_t *coords)

```

**lv\_draw\_ppa\_private.h****Defines****LV\_PPA\_NONBLOCKING\_OPS****PPA\_ALIGN\_UP(x, align)****PPA\_PTR\_ALIGN\_UP(p, align)****PPA\_ALIGN\_DOWN(x, align)****PPA\_PTR\_ALIGN\_DOWN(p, align)****Typedef**typedef struct *lv\_draw\_ppa\_unit* **lv\_draw\_ppa\_unit\_t****Functions**static inline bool **ppa\_src\_cf\_supported**(*lv\_color\_format\_t* cf)static inline bool **ppa\_dest\_cf\_supported**(*lv\_color\_format\_t* cf)static inline ppa\_fill\_color\_mode\_t **lv\_color\_format\_to\_ppa\_fill**(*lv\_color\_format\_t* lv\_fmt)static inline ppa\_blend\_color\_mode\_t **lv\_color\_format\_to\_ppa\_blend**(*lv\_color\_format\_t* lv\_fmt)static inline ppa\_srm\_color\_mode\_t **lv\_color\_format\_to\_ppa\_srm**(*lv\_color\_format\_t* lv\_fmt)struct **lv\_draw\_ppa\_unit****Public Members****lv\_draw\_unit\_t base\_unit****lv\_draw\_task\_t \*task\_act****ppa\_client\_handle\_t srm\_client****ppa\_client\_handle\_t fill\_client****ppa\_client\_handle\_t blend\_client****uint8\_t \*buf****14.12.25 eve****lv\_draw\_eve.h****Functions**void **lv\_draw\_eve\_init**(void)void **lv\_draw\_eve\_set\_display\_data**(*lv\_display\_t* \*disp, const *lv\_draw\_eve\_parameters\_t* \*params,  
*lv\_draw\_eve\_operation\_cb\_t* op\_cb)

**lv\_draw\_eve\_private.h****Defines****DEGREES(n)****F16(x)****lv\_draw\_eve\_unit\_g****Functions**

```
void lv_draw_eve_image(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t*coords)

void lv_draw_eve_fill(lv_draw_task_t *t, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)

void lv_draw_eve_border(lv_draw_task_t *t, const lv_draw_border_dsc_t *dsc, const lv_area_t *coords)

void lv_draw_eve_line(lv_draw_task_t *t, const lv_draw_line_dsc_t *dsc)

void lv_draw_eve_label(lv_draw_task_t *t, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords)

void lv_draw_eve_arc(lv_draw_task_t *t, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)

void lv_draw_eve_triangle(lv_draw_task_t *t, const lv_draw_triangle_dsc_t *dsc)
```

struct **lv\_draw\_eve\_ramg\_hash\_table\_cell\_t****Public Members**uintptr\_t **key**uint32\_t **addr**struct **lv\_draw\_eve\_ramg\_t****Public Members**uint32\_t **ramg\_addr\_end**uint32\_t **hash\_table\_cell\_count**uint32\_t **hash\_table\_cells\_occupied***lv\_draw\_eve\_ramg\_hash\_table\_cell\_t* \***hash\_table**struct **\_lv\_draw\_eve\_unit\_t**

**Public Members**

`lv_draw_unit_t base_unit`

`lv_draw_task_t *task_act`

`lv_display_t *disp`

`lv_draw_eve_ramg_t ramg`

`lv_draw_eve_parameters_t params`

`lv_draw_eve_operation_cb_t op_cb`

**lv\_draw\_eve\_ram\_g.h****Defines**

`LV_DRAW_EVE_RAMG_OUT_OF_RAMG`

**Functions**

`bool lv_draw_eve_ramg_get_addr(uint32_t *addr_dst, uintptr_t key, uint32_t addr_size, uint32_t addr_align)`

**lv\_draw\_eve\_target.h****Typedef**

`typedef void (*lv_draw_eve_operation_cb_t)(lv_display_t *disp, lv_draw_eve_operation_t operation, void *data, uint32_t length)`

**Enums**

`enum lv_draw_eve_operation_t`

*Values:*

enumerator `LV_DRAW_EVE_OPERATION_POWERDOWN_SET`

set the "PD\_N" pin low

enumerator `LV_DRAW_EVE_OPERATION_POWERDOWN_CLEAR`

set the "PD\_N" pin high

enumerator `LV_DRAW_EVE_OPERATION_CS_ASSERT`

set the "CS\_N" pin low

enumerator `LV_DRAW_EVE_OPERATION_CS_DEASSERT`

set the "CS\_N" pin high

enumerator `LV_DRAW_EVE_OPERATION_SPI_SEND`

send `length` bytes of `data` over SPI

**enumerator LV\_DRAW\_EVE\_OPERATION\_SPI\_RECEIVE**

receive `length` bytes into `data` from SPI

struct **lv\_draw\_eve\_parameters\_t**

**Public Members****uint16\_t hor\_res**

active display width

**uint16\_t ver\_res**

active display height

**uint16\_t hcycle**

total number of clocks per line, incl front/back porch

**uint16\_t hoffset**

start of active line

**uint16\_t hsync0**

start of horizontal sync pulse

**uint16\_t hsync1**

end of horizontal sync pulse

**uint16\_t vcycle**

total number of lines per screen, including pre/post

**uint16\_t voffset**

start of active screen

**uint16\_t vsync0**

start of vertical sync pulse

**uint16\_t vsync1**

end of vertical sync pulse

**uint8\_t swizzle**

FT8xx output to LCD - pin order

**uint8\_t pclkpol**

LCD data is clocked in on this PCLK edge

**uint8\_t cspread**

helps with noise, when set to 1 fewer signals are changed simultaneously, reset-default: 1

**uint8\_t pclk**

60MHz / pclk = pclk frequency

```

bool has_crystal
    has an external clock crystal

bool has_gt911
    has a touch controller

uint8_t backlight_pwm
    backlight PWM duty cycle 0 = off, 128 = max

uint16_t backlight_freq
    backlight PWM frequency. try 4000 if unsure

```

## lv\_eve.h

### Enums

enum **lv\_eve\_primitive\_t**

*Values:*

enumerator **LV\_EVE\_PRIMITIVE\_ZERO\_VALUE**

enumerator **LV\_EVE\_PRIMITIVE\_BITMAPS**

enumerator **LV\_EVE\_PRIMITIVE\_POINTS**

enumerator **LV\_EVE\_PRIMITIVE\_LINES**

enumerator **LV\_EVE\_PRIMITIVE\_LINE\_STRIP**

enumerator **LV\_EVE\_PRIMITIVE\_EDGE\_STRIP\_R**

enumerator **LV\_EVE\_PRIMITIVE\_EDGE\_STRIP\_L**

enumerator **LV\_EVE\_PRIMITIVE\_EDGE\_STRIP\_A**

enumerator **LV\_EVE\_PRIMITIVE\_EDGE\_STRIP\_B**

enumerator **LV\_EVE\_PRIMITIVE\_RECTS**

### Functions

void **lv\_eve\_save\_context**(void)

void **lv\_eve\_restore\_context**(void)

void **lv\_eve\_scissor**(uint16\_t x1, uint16\_t y1, uint16\_t x2, uint16\_t y2)

void **lv\_eve\_primitive**(uint8\_t context)

void **lv\_eve\_color**(*lv\_color\_t* color)

```

void lv_eve_color_opa(lv_opa_t opa)
void lv_eve_line_width(int32_t width)
void lv_eve_point_size(uint16_t radius)
void lv_eve_vertex_2f(int16_t x, int16_t y)
void lv_eve_color_mask(uint8_t r, uint8_t g, uint8_t b, uint8_t a)
void lv_eve_stencil_func(uint8_t func, uint8_t ref, uint8_t mask)
void lv_eve_stencil_op(uint8_t sfail, uint8_t spass)
void lv_eve_blend_func(uint8_t sfail, uint8_t spass)
void lv_eve_draw_circle_simple(int16_t coord_x1, int16_t coord_y1, uint16_t radius_t)
void lv_eve_draw_rect_simple(int16_t coord_x1, int16_t coord_y1, int16_t coord_x2, int16_t coord_y2,
                           uint16_t radius)
void lv_eve_mask_round(int16_t coord_x1, int16_t coord_y1, int16_t coord_x2, int16_t coord_y2, int16_t
                       radius)

struct lv_eve_drawing_context_t

```

#### Public Members

*lv\_eve\_primitive\_t* **primitive**

*lv\_color\_t* **color**

lv\_opa\_t **opa**

int32\_t **line\_width**

uint16\_t **point\_size**

uint8\_t **color\_mask**[4]

uint8\_t **stencil\_func**[3]

uint8\_t **stencil\_op**[2]

uint8\_t **blend\_func**[2]

uint16\_t **scx**

uint16\_t **scy**

## 14.12.26 nema\_gfx

**lv\_draw\_nema\_gfx.h**

MIT License

*Copyright (c) 2008-24 Think Silicon Single Member PC*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice (including the next paragraph) shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### Functions

```
void lv_draw_nema_gfx_init(void)
void lv_draw_nema_gfx_deinit(void)
void lv_draw_nema_gfx_fill(lv_draw_task_t *t, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_nema_gfx_triangle(lv_draw_task_t *t, const lv_draw_triangle_dsc_t *dsc)
void lv_draw_nema_gfx_img(lv_draw_task_t *t, const lv_draw_image_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_nema_gfx_label(lv_draw_task_t *t, const lv_draw_label_dsc_t *dsc, const lv_area_t
                           *coords)
void lv_draw_nema_gfx_label_init(lv_draw_unit_t *draw_unit)
void lv_draw_nema_gfx_layer(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t
                           *coords)
void lv_draw_nema_gfx_line(lv_draw_task_t *t, const lv_draw_line_dsc_t *dsc)
void lv_draw_nema_gfx_border(lv_draw_task_t *t, const lv_draw_border_dsc_t *dsc, const lv_area_t
                           *coords)
void lv_draw_nema_gfx_arc(lv_draw_task_t *t, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)

struct lv_draw_nema_gfx_unit_t
```

### Public Members

lv\_draw\_unit\_t **base\_unit**

lv\_draw\_task\_t \***task\_act**

uint32\_t **idx**

`nema_cmdlist_t cl`

`NEMA_VG_PAINT_HANDLE paint`

`NEMA_VG_GRAD_HANDLE gradient`

`NEMA_VG_PATH_HANDLE path`

## lv\_draw\_nema\_gfx\_utils.h

MIT License

*Copyright (c) 2008-24 Think Silicon Single Member PC*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice (including the next paragraph) shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

### Defines

`NEMA_VIRT2PHYS`

`LV_NEMA_GFX_COLOR_FORMAT`

`LV_NEMA_GFX_FORMAT_MULTIPLIER`

### Functions

`bool lv_nemagfx_is_cf_supported(lv_color_format_t cf)`

Check if `lv_color_format_t` is supported.

**Parametri**

`cf` -- The LVGL color format

**Ritorna**

True/false

`uint32_t lv_nemagfx_cf_to_nema(lv_color_format_t cf)`

Convert a `lv_color_format_t` to a Nema color format.

**Parametri**

`cf` -- The LVGL color format

**Ritorna**

The Nema color format

`uint32_t lv_nemagfx_blending_mode(lv_blend_mode_t lv_blend_mode)`

Get NemaGFX blending mode

**Parametri**

`lv_blend_mode` -- [in] The LVGL blend mode

**Ritorna**

NemaGFX blending mode

```
void lv_nemagfx_grad_set(NEMA_VG_GRAD_HANDLE gradient, lv_grad_dsc_t lv_grad, lv_opa_t opa)
```

Get NemaGFX blending mode

**Parametri**

- **gradient** -- [in] NemaGFX Gradient Buffer
- **lv\_grad** -- [in] Gradient descriptor
- **opa** -- [in] Descriptor's opacity

**lv\_nema\_gfx\_path.h**

MIT License

*Copyright (c) 2008-24 Think Silicon Single Member PC*

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice (including the next paragraph) shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

**Functions**

```
lv_nema_gfx_path_t *lv_nema_gfx_path_create(void)
void lv_nema_gfx_path_alloc(lv_nema_gfx_path_t *nema_gfx_path)
void lv_nema_gfx_path_destroy(lv_nema_gfx_path_t *nema_gfx_path)
void lv_nema_gfx_path_move_to(lv_nema_gfx_path_t *nema_gfx_path, float x, float y)
void lv_nema_gfx_path_line_to(lv_nema_gfx_path_t *nema_gfx_path, float x, float y)
void lv_nema_gfx_path_quad_to(lv_nema_gfx_path_t *nema_gfx_path, float cx, float cy, float x, float y)
void lv_nema_gfx_path_cubic_to(lv_nema_gfx_path_t *nema_gfx_path, float cx1, float cy1, float cx2,
                               float cy2, float x, float y)
void lv_nema_gfx_path_end(lv_nema_gfx_path_t *nema_gfx_path)
```

struct **lv\_nema\_gfx\_path\_t**

**Public Members**

NEMA\_VG\_PATH\_HANDLE **path**

NEMA\_VG\_PAINT\_HANDLE **paint**

```
float *data  
uint8_t *seg  
uint32_t data_size  
uint32_t seg_size
```

### 14.12.27 nxp

**g2d**

**lv\_draw\_g2d.h**

#### Typedef

typedef struct *lv\_draw\_g2d\_unit* **lv\_draw\_g2d\_unit\_t**

Copyright 2024 NXP

SPDX-License-Identifier: MIT

#### Functions

void **lv\_draw\_g2d\_init**(void)

void **lv\_draw\_g2d\_deinit**(void)

void **lv\_draw\_buf\_g2d\_init\_handlers**(void)

void **lv\_draw\_g2d\_fill**(lv\_draw\_task\_t \*t)

void **lv\_draw\_g2d\_img**(lv\_draw\_task\_t \*t)

struct **lv\_draw\_g2d\_unit**

#include <lv\_draw\_g2d.h> Copyright 2024 NXP

SPDX-License-Identifier: MIT

#### Public Members

lv\_draw\_unit\_t **base\_unit**

lv\_draw\_task\_t \***task\_act**

void \***g2d\_handle**

**lv\_g2d\_buf\_map.h**

#### Typedef

typedef struct *lv\_map\_item* **lv\_map\_item\_t**

Copyright 2024 NXP

SPDX-License-Identifier: MIT

typedef struct *lv\_buf\_map* **lv\_buf\_map\_t**

**Functions**

```
void g2d_create_buf_map(void)
void g2d_free_buf_map(void)
void g2d_insert_buf_map(void *key, struct g2d_buf *value)
struct g2d_buf *g2d_search_buf_map(void *key)
void g2d_free_item(void *key)
void g2d_print_table(void)
```

struct **lv\_map\_item**

#include <lv\_g2d\_buf\_map.h> Copyright 2024 NXP

SPDX-License-Identifier: MIT

**Public Members**

void \***key**

struct g2d\_buf \***value**

struct **lv\_buf\_map**

**Public Members**

*lv\_map\_item\_t* \*\***items**

lv\_array\_t \*\***overflow\_list**

int **size**

int **count**

**lv\_g2d\_utils.h**

**Defines**

**G2D\_ASSERT**(expr)

Copyright 2024 NXP

SPDX-License-Identifier: MIT

**G2D\_ASSERT\_MSG**(expr, msg)

**Typedef**

typedef enum g2d\_format **g2d\_format\_t**

**Functions**

`g2d_format_t g2d_get_buf_format(lv_color_format_t cf)`

`uint32_t g2d_rgba_to_u32(lv_color_t color)`

`int32_t g2d_get_buf_fd(const lv_draw_buf_t *draw_buf)`

**pxp**

**lv\_draw\_pxp.h**

**Typedef**

`typedef struct _lv_draw_pxp_unit_t lv_draw_pxp_unit_t`

Copyright 2022-2024 NXP

SPDX-License-Identifier: MIT

**Functions**

`void lv_draw_pxp_init(void)`

`void lv_draw_pxp_deinit(void)`

`void lv_draw_pxp_rotate(const void *src_buf, void *dest_buf, int32_t src_width, int32_t src_height,  
int32_t src_stride, int32_t dest_stride, lv_display_rotation_t rotation,  
lv_color_format_t cf)`

`void lv_draw_buf_pxp_init_handlers(void)`

`void lv_draw_pxp_fill(lv_draw_task_t *t)`

`void lv_draw_pxp_img(lv_draw_task_t *t)`

`void lv_draw_pxp_layer(lv_draw_task_t *t)`

`struct _lv_draw_pxp_unit_t`

`#include <lv_draw_pxp.h>` Copyright 2022-2024 NXP

SPDX-License-Identifier: MIT

**Public Members**

`lv_draw_unit_t base_unit`

`lv_draw_task_t *task_act`

**lv\_pxp\_cfg.h**

**Defines****PXP\_ID**

Copyright 2020-2023 NXP

SPDX-License-Identifier: MIT PXP module instance to use

**PXP\_IRQ\_ID**

PXP interrupt line ID

## Functions

`void lv_pxp_init(void)`

Reset and initialize PXP device. This function should be called as a part of display init sequence.

`void lv_pxp_deinit(void)`

Disable PXP device. Should be called during display deinit sequence.

`void lv_pxp_reset(void)`

Reset PXP device.

`void lv_pxp_run(void)`

Clear cache and start PXP.

`void lv_pxp_wait(void)`

Wait for PXP completion.

`struct ppxp_cfg_t`

`#include <lv_pxp_cfg.h>` NXP PXP device configuration.

## Public Members

`void (*ppxp_interrupt_init)(void)`

Callback for PXP interrupt initialization

`void (*ppxp_interrupt_deinit)(void)`

Callback for PXP interrupt de-initialization

`void (*ppxp_run)(void)`

Callback for PXP start

`void (*ppxp_wait)(void)`

Callback for waiting of PXP completion

## lv\_pxp\_osa.h

### Functions

`void PXP_IRQHandler(void)`

Copyright 2020, 2022-2023 NXP

SPDX-License-Identifier: MIT PXP device interrupt handler. Used to check PXP task completion status.

`ppxp_cfg_t *ppxp_get_default_cfg(void)`

Get the PXP default configuration.

## lv\_pxp\_utils.h

### Defines

`PXP_ASSERT(expr)`

Copyright 2023-2024 NXP

SPDX-License-Identifier: MIT

`PXP_ASSERT_MSG(expr, msg)`

**Functions**

```
pxp_output_pixel_format_t pxp_get_out_px_format(lv_color_format_t cf)
pxp_as_pixel_format_t pxp_get_as_px_format(lv_color_format_t cf)
pxp_ps_pixel_format_t pxp_get_ps_px_format(lv_color_format_t cf)
bool pxp_buf_aligned(const void *buf, uint32_t stride)
```

**vglite****lv\_draw\_vglite.h****Typedef**

```
typedef struct vglite_draw_task vglite_draw_task_t
```

Copyright 2023-2024 NXP

SPDX-License-Identifier: MIT

```
typedef struct lv_draw_vglite_unit lv_draw_vglite_unit_t
```

**Functions**

```
void lv_draw_buf_vglite_init_handlers(void)
void lv_draw_vglite_init(void)
void lv_draw_vglite_deinit(void)
void lv_draw_vglite_arc(vglite_draw_task_t *vglite_task)
void lv_draw_vglite_border(vglite_draw_task_t *vglite_task)
void lv_draw_vglite_fill(vglite_draw_task_t *vglite_task)
void lv_draw_vglite_img(vglite_draw_task_t *vglite_task)
void lv_draw_vglite_label(vglite_draw_task_t *vglite_task)
void lv_draw_vglite_layer(vglite_draw_task_t *vglite_task)
void lv_draw_vglite_line(vglite_draw_task_t *vglite_task)
void lv_draw_vglite_triangle(vglite_draw_task_t *vglite_task)
```

**struct vglite\_draw\_task**

#include <lv\_draw\_vglite.h> Copyright 2023-2024 NXP

SPDX-License-Identifier: MIT

**Public Members**

*lv\_draw\_task\_t* \***t**

*vg\_lite\_path\_t* \***path**

*vg\_lite\_linear\_gradient\_t* \***gradient**

```
int32_t *path_data

struct lv_draw_vglite_unit
```

### Public Members

lv\_draw\_unit\_t **base\_unit**

*vglite\_draw\_task\_t* \***task\_act**

**lv\_vglite\_buf.h**

### Functions

vg\_lite\_buffer\_t \***vglite\_get\_dest\_buf**(void)

Copyright 2023 NXP

SPDX-License-Identifier: MIT Get vglite destination buffer pointer.

#### Valori restituiti

**The** -- vglite destination buffer

vg\_lite\_buffer\_t \***vglite\_get\_src\_buf**(void)

Get vglite source buffer pointer.

#### Valori restituiti

**The** -- vglite source buffer

void **vglite\_set\_dest\_buf\_ptr**(void \*buf)

Set vglite destination buffer address only.

#### Parametri

- **buf** -- [in] Destination buffer address (does not require alignment for VG\_LITE\_LINEAR mode)

void **vglite\_set\_src\_buf\_ptr**(const void \*buf)

Set vglite source buffer address only.

#### Parametri

- **buf** -- [in] Source buffer address

void **vglite\_set\_dest\_buf**(const void \*buf, uint32\_t width, uint32\_t height, uint32\_t stride,  
*lv\_color\_format\_t* cf)

Set vglite destination buffer.

#### Parametri

- **buf** -- [in] Destination buffer address
- **width** -- [in] Destination buffer width
- **height** -- [in] Destination buffer height
- **stride** -- [in] Destination buffer stride in bytes
- **cf** -- [in] Destination buffer color format

void **vglite\_set\_src\_buf**(const void \*buf, uint32\_t width, uint32\_t height, uint32\_t stride,  
*lv\_color\_format\_t* cf)

Set vglite source buffer.

#### Parametri

- **buf** -- [in] Source buffer address
- **width** -- [in] Source buffer width
- **height** -- [in] Source buffer height
- **stride** -- [in] Source buffer stride in bytes
- **cf** -- [in] Source buffer color format

---

```
void vglite_set_buf(vg_lite_buffer_t *vdbuf, void *buf, uint32_t width, uint32_t height, uint32_t stride,
                      lv_color_format_t cf)
```

Set vglite buffer.

#### Parametri

- **vdbuf** -- [in] Address of the VGLite buffer object
- **buf** -- [in] Address of the memory for the VGLite buffer
- **width** -- [in] Buffer width
- **height** -- [in] Buffer height
- **stride** -- [in] Buffer stride in bytes
- **cf** -- [in] Buffer color format

## lv\_vglite\_matrix.h

### Functions

```
vg_lite_matrix_t *vglite_get_matrix(void)
```

Copyright 2023 NXP

SPDX-License-Identifier: MIT

```
void vglite_set_translation_matrix(const lv_area_t *dest_area)
```

Creates matrix that translates to origin of given destination area.

#### Parametri

- **dest\_area** -- [in] Area with relative coordinates of destination buffer

```
void vglite_set_transformation_matrix(const lv_area_t *dest_area, const lv_draw_image_dsc_t
                                         *dsc)
```

Creates matrix that translates to origin of given destination area with transformation (scale or rotate).

#### Parametri

- **dest\_area** -- [in] Area with relative coordinates of destination buffer
- **dsc** -- [in] Image descriptor

## lv\_vglite\_path.h

### Defines

**BEZIER\_OPTIM\_CIRCLE**

Copyright 2023 NXP

SPDX-License-Identifier: MIT

**BEZIER\_DBG\_CONTROL\_POINTS**

**CUBIC\_PATH\_DATA\_SIZE**

**LINE\_PATH\_DATA\_SIZE**

**MOVE\_PATH\_DATA\_SIZE**

**END\_PATH\_DATA\_SIZE**

**RECT\_PATH\_DATA\_MAX\_SIZE**

**ARC\_PATH\_DATA\_MAX\_SIZE**

## Functions

```
void vglite_create_rect_path_data(int32_t *path_data, uint32_t *path_data_size, int32_t radius,
                                const lv_area_t *coords)
```

Generates path data for rectangle drawing.

### Parametri

- **[in/out]** -- *path* The path data to initialize
- **[in/out]** -- *path\_size* The resulting size of the created path data
- **dsc** -- **[in]** The style descriptor for the rectangle to be drawn
- **coords** -- **[in]** The coordinates of the rectangle to be drawn

## *lv\_vglite\_utils.h*

### Defines

```
ENUM_TO_STRING(e)
```

Copyright 2022-2024 NXP

SPDX-License-Identifier: MIT

```
VGLITE_ASSERT(expr)
```

```
VGLITE_ASSERT_MSG(expr, msg)
```

```
VGLITE_CHECK_ERROR(function)
```

## Functions

```
static inline void vglite_set_scissor(const lv_area_t *clip_area)
```

Set the clipping box.

### Parametri

**clip\_area** -- **[in]** Clip area with relative coordinates of destination buffer

```
const char *vglite_error_to_string(vg_lite_error_t error)
```

```
void vglite_run(void)
```

Flush command to VG-Lite.

```
vg_lite_color_t vglite_get_color(lv_color32_t lv_col32, bool gradient)
```

Wait for VG-Lite finish. Get vglite color. Premultiplies (if not hw already) and swizzles the given LVGL 32bit color to obtain vglite color.

### Parametri

- **lv\_col32** -- **[in]** The initial LVGL 32bit color
- **gradient** -- **[in]** True for gradient color

### Valori restituiti

**The** -- vglite 32-bit color value:

```
vg_lite_blend_t vglite_get_blend_mode(lv_blend_mode_t lv_blend_mode)
```

Get vglite blend mode.

### Parametri

**lv\_blend\_mode** -- **[in]** The LVGL blend mode

### Valori restituiti

**The** -- vglite blend mode

```
vg_lite_buffer_format_t vglite_get_buf_format(lv_color_format_t cf)
```

Get vglite buffer format.

### Parametri

**cf** -- **[in]** Color format

### Valori restituiti

**The** -- vglite buffer format

`uint8_t vglite_get_stride_alignment(lv_color_format_t cf)`

Get vglite stride alignment.

**Parametri**

- **cf** -- [in] Color format

**Valori restituiti**

- **Alignment** -- requirement in bytes

`bool vglite_src_buf_aligned(const void *buf, uint32_t stride, lv_color_format_t cf)`

Check source start address and stride alignment.

**Parametri**

- **buf** -- [in] Buffer address
- **stride** -- [in] Stride of buffer in bytes
- **cf** -- [in] Color format - to calculate the expected alignment

**Valori restituiti**

- **true** -- Alignment OK

## 14.12.28 opengles

**lv\_draw\_opengles.h**

**Functions**

`void lv_draw_opengles_init(void)`

`void lv_draw_opengles_deinit(void)`

## 14.12.29 renesas

**dave2d**

**lv\_draw\_dave2d.h**

**Defines**

**D2\_RENDER\_EACH\_OPERATION**

**Functions**

`void lv_draw_dave2d_init(void)`

`void lv_draw_dave2d_image(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)`

`void lv_draw_dave2d_fill(lv_draw_task_t *t, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)`

`void lv_draw_dave2d_border(lv_draw_task_t *t, const lv_draw_border_dsc_t *dsc, const lv_area_t *coords)`

`void lv_draw_dave2d_box_shadow(lv_draw_task_t *t, const lv_draw_box_shadow_dsc_t *dsc, const lv_area_t *coords)`

`void lv_draw_dave2d_label(lv_draw_task_t *t, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords)`

`void lv_draw_dave2d_arc(lv_draw_task_t *t, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)`

`void lv_draw_dave2d_line(lv_draw_task_t *t, const lv_draw_line_dsc_t *dsc)`

`void lv_draw_dave2d_layer(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)`

```
void lv_draw_dave2d_triangle(lv_draw_task_t *t, const lv_draw_triangle_dsc_t *dsc)
void lv_draw_dave2d_mask_rect(lv_draw_task_t *t, const lv_draw_mask_rect_dsc_t *dsc, const
                             lv_area_t *coords)

void lv_draw_dave2d_transform(lv_draw_task_t *t, const lv_area_t *dest_area, const void *src_buf,
                            int32_t src_w, int32_t src_h, int32_t src_stride, const
                            lv_draw_image_dsc_t *draw_dsc, const lv_draw_image_sup_t *sup,
                            lv_color_format_t cf, void *dest_buf)
```

struct **lv\_draw\_dave2d\_unit\_t**

#### Public Members

```
lv_draw_unit_t base_unit
lv_draw_task_t *task_act
uint32_t idx
d2_device *d2_handle
d2_renderbuffer *renderbuffer
```

**lv\_draw\_dave2d\_utils.h**

#### Functions

```
d2_color lv_draw_dave2d_lv_colour_to_d2_colour(lv_color_t color)
d2_s32 lv_draw_dave2d_cf_fb_get(void)
d2_u32 lv_draw_dave2d_lv_colour_fmt_to_d2_fmt(lv_color_format_t colour_format)
void d2_framebuffer_from_layer(d2_device *handle, lv_layer_t *layer)
bool lv_draw_dave2d_is_dest_cf_supported(lv_color_format_t cf)
```

### 14.12.30 sdl

**lv\_draw\_sdl.h**

#### Functions

```
void lv_draw_sdl_init(void)
void lv_draw_sdl_image(lv_draw_unit_t *draw_unit, const lv_draw_image_dsc_t *draw_dsc, const
                      lv_area_t *coords)
void lv_draw_sdl_fill(lv_draw_unit_t *draw_unit, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)
void lv_draw_sdl_border(lv_draw_unit_t *draw_unit, const lv_draw_border_dsc_t *dsc, const lv_area_t
                       *coords)
void lv_draw_sdl_box_shadow(lv_draw_unit_t *draw_unit, const lv_draw_box_shadow_dsc_t *dsc, const
                           lv_area_t *coords)
```

```
void lv_draw_sdl_label(lv_draw_unit_t *draw_unit, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords)

void lv_draw_sdl_arc(lv_draw_unit_t *draw_unit, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)

void lv_draw_sdl_line(lv_draw_unit_t *draw_unit, const lv_draw_line_dsc_t *dsc)

void lv_draw_sdl_layer(lv_draw_unit_t *draw_unit, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)

void lv_draw_sdl_triangle(lv_draw_unit_t *draw_unit, const lv_draw_triangle_dsc_t *dsc)

void lv_draw_sdl_mask_rect(lv_draw_unit_t *draw_unit, const lv_draw_mask_rect_dsc_t *dsc, const lv_area_t *coords)
```

### 14.12.31 sw

#### lv\_draw\_sw.h

##### Functions

void **lv\_draw\_sw\_init**(void)

Initialize the SW renderer. Called in internally. It creates as many SW renderers as defined in LV\_DRAW\_SW\_DRAW\_UNIT\_CNT

void **lv\_draw\_sw\_deinit**(void)

Deinitialize the SW renderers

void **lv\_draw\_sw\_fill**(lv\_draw\_task\_t \*t, lv\_draw\_fill\_dsc\_t \*dsc, const lv\_area\_t \*coords)

Fill an area using SW render. Handle gradient and radius.

##### Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor
- **coords** -- the coordinates of the rectangle

void **lv\_draw\_sw\_border**(lv\_draw\_task\_t \*t, const lv\_draw\_border\_dsc\_t \*dsc, const lv\_area\_t \*coords)

Draw border with SW render.

##### Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor
- **coords** -- the coordinates of the rectangle

void **lv\_draw\_sw\_box\_shadow**(lv\_draw\_task\_t \*t, const lv\_draw\_box\_shadow\_dsc\_t \*dsc, const lv\_area\_t \*coords)

Draw box shadow with SW render.

##### Parametri

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor
- **coords** -- the coordinates of the rectangle for which the box shadow should be drawn

void **lv\_draw\_sw\_image**(lv\_draw\_task\_t \*t, const lv\_draw\_image\_dsc\_t \*draw\_dsc, const lv\_area\_t \*coords)

Draw an image with SW render. It handles image decoding, tiling, transformations, and recoloring.

##### Parametri

- **t** -- pointer to a draw task
- **draw\_dsc** -- the draw descriptor
- **coords** -- the coordinates of the image

void **lv\_draw\_sw\_letter**(lv\_draw\_task\_t \*t, const lv\_draw\_letter\_dsc\_t \*dsc, const lv\_area\_t \*coords)

---

```
void lv_draw_sw_label(lv_draw_task_t *t, const lv_draw_label_dsc_t *dsc, const lv_area_t *coords)
```

Draw a label with SW render.

**Parametri**

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor
- **coords** -- the coordinates of the label

```
void lv_draw_sw_arc(lv_draw_task_t *t, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)
```

Draw an arc with SW render.

**Parametri**

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor
- **coords** -- the coordinates of the arc

```
void lv_draw_sw_line(lv_draw_task_t *t, const lv_draw_line_dsc_t *dsc)
```

Draw a line with SW render.

**Parametri**

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor

```
void lv_draw_sw_layer(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t *coords)
```

Blend a layer with SW render

**Parametri**

- **t** -- pointer to a draw task
- **draw\_dsc** -- the draw descriptor
- **coords** -- the coordinates of the layer

```
void lv_draw_sw_triangle(lv_draw_task_t *t, const lv_draw_triangle_dsc_t *dsc)
```

Draw a triangle with SW render.

**Parametri**

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor

```
void lv_draw_sw_mask_rect(lv_draw_task_t *t, const lv_draw_mask_rect_dsc_t *dsc)
```

Mask out a rectangle with radius from a current layer

**Parametri**

- **t** -- pointer to a draw task
- **dsc** -- the draw descriptor
- **coords** -- the coordinates of the mask

```
void lv_draw_sw_transform(const lv_area_t *dest_area, const void *src_buf, int32_t src_w, int32_t src_h, int32_t src_stride, const lv_draw_image_dsc_t *draw_dsc, const lv_draw_image_sup_t *sup, lv_color_format_t cf, void *dest_buf)
```

Used internally to get a transformed area of an image

**Parametri**

- **dest\_area** -- area to calculate, i.e. get this area from the transformed image
- **src\_buf** -- source buffer
- **src\_w** -- source buffer width in pixels
- **src\_h** -- source buffer height in pixels
- **src\_stride** -- source buffer stride in bytes
- **draw\_dsc** -- draw descriptor
- **sup** -- supplementary data
- **cf** -- color format of the source buffer
- **dest\_buf** -- the destination buffer

```
void lv_draw_sw_vector(lv_draw_task_t *t, lv_draw_vector_task_dsc_t *dsc)
```

Draw vector graphics with SW render.

**Parametri**

- **t** -- pointer to a draw task

- **dsc** -- the draw descriptor

**bool lv\_draw\_sw\_register\_blend\_handler(*lv\_draw\_sw\_custom\_blend\_handler\_t* \*handler)**

Register a custom blend handler for a color format. Handler will be called when blending a color or an image to a buffer with the given color format. At most one handler can be registered for a color format. Subsequent registrations will overwrite the previous handler.

**Parametri**

**handler** -- pointer to a blend handler

**Ritorna**

true if the handler was registered, false if the handler could not be registered

**bool lv\_draw\_sw\_unregister\_blend\_handler(*lv\_color\_format\_t* dest\_cf)**

Unregister a custom blend handler for a color format.

**Parametri**

**dest\_cf** -- color format

**Ritorna**

true if a handler was unregistered, false if no handler was registered

*lv\_draw\_sw\_blend\_handler\_t* **lv\_draw\_sw\_get\_blend\_handler(*lv\_color\_format\_t* dest\_cf)**

Get the blend handler for a color format.

**Parametri**

**dest\_cf** -- color format

**Ritorna**

pointer to the blend handler or NULL if no handler is registered

## lv\_draw\_sw\_grad.h

### Functions

**void lv\_draw\_sw\_grad\_color\_calculate(const *lv\_grad\_dsc\_t* \*dsc, int32\_t range, int32\_t frac,  
  *lv\_color\_t* \*color\_out, *lv\_opa\_t* \*opa\_out)**

Compute the color in the given gradient and fraction Gradient are specified in a virtual [0-255] range, so this function scales the virtual range to the given range

**Parametri**

- **dsc** -- The gradient descriptor to use
- **range** -- The range to use in computation.
- **frac** -- The current part used in the range. frac is in [0; range]
- **color\_out** -- Calculated gradient color
- **opa\_out** -- Calculated opacity

*lv\_draw\_sw\_grad\_calc\_t* \***lv\_draw\_sw\_grad\_get(const *lv\_grad\_dsc\_t* \*gradient, int32\_t w, int32\_t h)**

Get a gradient cache from the given parameters

**void lv\_draw\_sw\_grad\_cleanup(*lv\_draw\_sw\_grad\_calc\_t* \*grad)**

Clean up the gradient item after it was get with *lv\_grad\_get\_from\_cache*.

**Parametri**

**grad** -- pointer to a gradient

**void lv\_draw\_sw\_grad\_linear\_setup(*lv\_grad\_dsc\_t* \*dsc, const *lv\_area\_t* \*coords)**

Calculate constants from the given parameters that are used during rendering

**Parametri**

- **dsc** -- gradient descriptor
- **coords** -- the area where to draw the gradient

**void lv\_draw\_sw\_grad\_linear\_cleanup(*lv\_grad\_dsc\_t* \*dsc)**

Free up the allocated memory for the gradient calculation

**Parametri**

**dsc** -- gradient descriptor

---

```
void lv_draw_sw_grad_linear_get_line(lv_grad_dsc_t *dsc, int32_t xp, int32_t yp, int32_t width,
                                     lv_draw_sw_grad_calc_t *result)
```

Calculate a line segment of a linear gradient

**Parametri**

- **dsc** -- gradient descriptor
- **xp** -- starting point x coordinate in gradient space
- **yp** -- starting point y coordinate in gradient space
- **width** -- width of the line segment in pixels
- **result** -- color buffer for the resulting line segment

```
void lv_draw_sw_grad_radial_setup(lv_grad_dsc_t *dsc, const lv_area_t *coords)
```

Calculate constants from the given parameters that are used during rendering

**Parametri**

- **dsc** -- gradient descriptor
- **coords** -- the area where to draw the gradient

```
void lv_draw_sw_grad_radial_cleanup(lv_grad_dsc_t *dsc)
```

Free up the allocated memory for the gradient calculation

**Parametri**

**dsc** -- gradient descriptor

```
void lv_draw_sw_grad_radial_get_line(lv_grad_dsc_t *dsc, int32_t xp, int32_t yp, int32_t width,
                                      lv_draw_sw_grad_calc_t *result)
```

Calculate a line segment of a radial gradient

**Parametri**

- **dsc** -- gradient descriptor
- **xp** -- starting point x coordinate in gradient space
- **yp** -- starting point y coordinate in gradient space
- **width** -- width of the line segment in pixels
- **result** -- color buffer for the resulting line segment

```
void lv_draw_sw_grad_conical_setup(lv_grad_dsc_t *dsc, const lv_area_t *coords)
```

Calculate constants from the given parameters that are used during rendering

**Parametri**

- **dsc** -- gradient descriptor
- **coords** -- the area where to draw the gradient

```
void lv_draw_sw_grad_conical_cleanup(lv_grad_dsc_t *dsc)
```

Free up the allocated memory for the gradient calculation

**Parametri**

**dsc** -- gradient descriptor

```
void lv_draw_sw_grad_conical_get_line(lv_grad_dsc_t *dsc, int32_t xp, int32_t yp, int32_t width,
                                       lv_draw_sw_grad_calc_t *result)
```

Calculate a line segment of a conical gradient

**Parametri**

- **dsc** -- gradient descriptor
- **xp** -- starting point x coordinate in gradient space
- **yp** -- starting point y coordinate in gradient space
- **width** -- width of the line segment in pixels
- **result** -- color buffer for the resulting line segment

---

```
struct lv_draw_sw_grad_calc_t
```

## Public Members

`lv_color_t *color_map`

`lv_opa_t *opa_map`

`uint32_t size`

**lv\_draw\_sw\_mask.h**

## Defines

**LV\_MASK\_ID\_INV**

**LV\_MASK\_MAX\_NUM**

## Typedef

typedef `lv_draw_sw_mask_res_t (*lv_draw_sw_mask_xcb_t)(lv_opa_t *mask_buf, int32_t abs_x, int32_t abs_y, int32_t len, void *p)`

A common callback type for every mask type. Used internally by the library.

## Enums

enum **lv\_draw\_sw\_mask\_res\_t**

*Values:*

enumerator **LV\_DRAW\_SW\_MASK\_RES\_TRANSP**

enumerator **LV\_DRAW\_SW\_MASK\_RES\_FULL\_COVER**

enumerator **LV\_DRAW\_SW\_MASK\_RES\_CHANGED**

enumerator **LV\_DRAW\_SW\_MASK\_RES\_UNKNOWN**

enum **lv\_draw\_sw\_mask\_type\_t**

*Values:*

enumerator **LV\_DRAW\_SW\_MASK\_TYPE\_LINE**

enumerator **LV\_DRAW\_SW\_MASK\_TYPE\_ANGLE**

enumerator **LV\_DRAW\_SW\_MASK\_TYPE\_RADIUS**

enumerator **LV\_DRAW\_SW\_MASK\_TYPE\_FADE**

enumerator **LV\_DRAW\_SW\_MASK\_TYPE\_MAP**

---

enum **lv\_draw\_sw\_mask\_line\_side\_t**

*Values:*

enumerator **LV\_DRAW\_SW\_MASK\_LINE\_SIDE\_LEFT**

enumerator **LV\_DRAW\_SW\_MASK\_LINE\_SIDE\_RIGHT**

enumerator **LV\_DRAW\_SW\_MASK\_LINE\_SIDE\_TOP**

enumerator **LV\_DRAW\_SW\_MASK\_LINE\_SIDE\_BOTTOM**

## Functions

void **lv\_draw\_sw\_mask\_init**(void)

void **lv\_draw\_sw\_mask\_deinit**(void)

*lv\_draw\_sw\_mask\_res\_t lv\_draw\_sw\_mask\_apply*(void \*masks[], lv\_opa\_t \*mask\_buf, int32\_t abs\_x,  
int32\_t abs\_y, int32\_t len)

Apply the added buffers on a line. Used internally by the library's drawing routines.

### Parametri

- **masks** -- the masks list to apply, must be ended with NULL pointer in array.
- **mask\_buf** -- store the result mask here. Has to be **len** byte long. Should be initialized with **0xFF**.
- **abs\_x** -- absolute X coordinate where the line to calculate start
- **abs\_y** -- absolute Y coordinate where the line to calculate start
- **len** -- length of the line to calculate (in pixel count)

### Ritorna

One of these values:

- **LV\_DRAW\_MASK\_RES\_FULL\_TRANSP**: the whole line is transparent. **mask\_buf** is not set to zero
- **LV\_DRAW\_MASK\_RES\_FULL\_COVER**: the whole line is fully visible. **mask\_buf** is unchanged
- **LV\_DRAW\_MASK\_RES\_CHANGED**: **mask\_buf** has changed, it shows the desired opacity of each pixel in the given line

void **lv\_draw\_sw\_mask\_free\_param**(void \*p)

Free the data from the parameter. It's called inside **lv\_draw\_sw\_mask\_remove\_id** and **lv\_draw\_sw\_mask\_remove\_custom**. Needs to be called only in special cases when the mask is not added by **lv\_draw\_mask\_add** and not removed by **lv\_draw\_mask\_remove\_id** or **lv\_draw\_mask\_remove\_custom**

### Parametri

**p** -- pointer to a mask parameter

void **lv\_draw\_sw\_mask\_line\_points\_init**(lv\_draw\_sw\_mask\_line\_param\_t \*param, int32\_t p1x,  
int32\_t p1y, int32\_t p2x, int32\_t p2y,  
*lv\_draw\_sw\_mask\_line\_side\_t* side)

Initialize a line mask from two points.

### Parametri

- **param** -- pointer to a **lv\_draw\_mask\_param\_t** to initialize
- **p1x** -- X coordinate of the first point of the line
- **p1y** -- Y coordinate of the first point of the line
- **p2x** -- X coordinate of the second point of the line
- **p2y** -- y coordinate of the second point of the line
- **side** -- an element of **lv\_draw\_sw\_mask\_line\_side\_t** to describe which side to keep. With **LV\_DRAW\_SW\_MASK\_LINE\_SIDE\_LEFT/RIGHT** and horizontal

line all pixels are kept With `LV_DRAW_MASK_LINE_SIDE_TOP/BOTTOM` and vertical line all pixels are kept

```
void lv_draw_sw_mask_line_angle_init(lv_draw_sw_mask_line_param_t *param, int32_t px, int32_t py, int16_t angle, lv_draw_sw_mask_line_side_t side)
```

Initialize a line mask from a point and an angle.

#### Parametri

- **param** -- pointer to a `lv_draw_mask_param_t` to initialize
- **px** -- X coordinate of a point of the line
- **py** -- Y coordinate of a point of the line
- **angle** -- right 0 deg, bottom: 90
- **side** -- an element of `lv_draw_mask_line_side_t` to describe which side to keep. With `LV_DRAW_MASK_LINE_SIDE_LEFT/RIGHT` and horizontal line all pixels are kept With `LV_DRAW_MASK_LINE_SIDE_TOP/BOTTOM` and vertical line all pixels are kept

```
void lv_draw_sw_mask_angle_init(lv_draw_sw_mask_angle_param_t *param, int32_t vertex_x, int32_t vertex_y, int32_t start_angle, int32_t end_angle)
```

Initialize an angle mask.

#### Parametri

- **param** -- pointer to a `lv_draw_mask_param_t` to initialize
- **vertex\_x** -- X coordinate of the angle vertex (absolute coordinates)
- **vertex\_y** -- Y coordinate of the angle vertex (absolute coordinates)
- **start\_angle** -- start angle in degrees. 0 deg on the right, 90 deg, on the bottom
- **end\_angle** -- end angle

```
void lv_draw_sw_mask_radius_init(lv_draw_sw_mask_radius_param_t *param, const lv_area_t *rect, int32_t radius, bool inv)
```

Initialize a fade mask.

#### Parametri

- **param** -- pointer to an `lv_draw_mask_radius_param_t` to initialize
- **rect** -- coordinates of the rectangle to affect (absolute coordinates)
- **radius** -- radius of the rectangle
- **inv** -- true: keep the pixels inside the rectangle; keep the pixels outside of the rectangle

```
void lv_draw_sw_mask_fade_init(lv_draw_sw_mask_fade_param_t *param, const lv_area_t *coords, lv_opa_t opa_top, int32_t y_top, lv_opa_t opa_bottom, int32_t y_bottom)
```

Initialize a fade mask.

#### Parametri

- **param** -- pointer to a `lv_draw_mask_param_t` to initialize
- **coords** -- coordinates of the area to affect (absolute coordinates)
- **opa\_top** -- opacity on the top
- **y\_top** -- at which coordinate start to change to opacity to **opa\_bottom**
- **opa\_bottom** -- opacity at the bottom
- **y\_bottom** -- at which coordinate reach **opa\_bottom**.

```
void lv_draw_sw_mask_map_init(lv_draw_sw_mask_map_param_t *param, const lv_area_t *coords, const lv_opa_t *map)
```

Initialize a map mask.

#### Parametri

- **param** -- pointer to a `lv_draw_mask_param_t` to initialize
- **coords** -- coordinates of the map (absolute coordinates)
- **map** -- array of bytes with the mask values

**lv\_draw\_sw\_mask\_private.h****Typedef**

```
typedef lv_draw_sw_mask_radius_circle_dsc_t lv_draw_sw_mask_radius_circle_dsc_arr_t[4]
```

**Functions**

```
void lv_draw_sw_mask_cleanup(void)
```

Called by LVGL the rendering of a screen is ready to clean up the temporal (cache) data of the masks

```
struct lv_draw_sw_mask_radius_circle_dsc_t
```

**Public Members**

**uint8\_t \*buf**

**lv\_opa\_t \*cir\_opa**

Opacity of values on the circumference of an 1/4 circle

**uint16\_t \*x\_start\_on\_y**

The x coordinate of the circle for each y value

**uint16\_t \*opa\_start\_on\_y**

The index of **cir\_opa** for each y value

**int32\_t life**

How many times the entry way used

**uint32\_t used\_cnt**

Like a semaphore to count the referencing masks

**int32\_t radius**

The radius of the entry

```
struct _lv_draw_sw_mask_common_dsc_t
```

**Public Members**

*lv\_draw\_sw\_mask\_xcb\_t cb*

*lv\_draw\_sw\_mask\_type\_t type*

```
struct _lv_draw_sw_mask_line_param_t
```

**Public Members**

**lv\_draw\_sw\_mask\_common\_dsc\_t dsc**

The first element must be the common descriptor

*lv\_point\_t* **p1**

*lv\_point\_t* **p2**

*lv\_draw\_sw\_mask\_line\_side\_t* **side**

struct *lv\_draw\_sw\_mask\_line\_param\_t* **cfg**

*lv\_point\_t* **origo**

A point of the line

int32\_t **xy\_stEEP**

X / (1024\*Y) steepness (X is 0..1023 range). What is the change of X in 1024 Y?

int32\_t **yx\_stEEP**

Y / (1024\*X) steepness (Y is 0..1023 range). What is the change of Y in 1024 X?

int32\_t **steEP**

Helper which stores yx\_stEEP for flat lines and xy\_stEEP for steep (non flat) lines

int32\_t **spx**

Steepness in 1 px in 0..255 range. Used only by flat lines.

uint8\_t **flat**

1: It's a flat line? (Near to horizontal)

uint8\_t **inv**

Invert the mask. The default is: Keep the left part. It is used to select left/right/top/bottom

struct *lv\_draw\_sw\_mask\_angle\_param\_t*

### Public Members

*lv\_draw\_sw\_mask\_common\_dsc\_t* **dsc**

The first element must be the common descriptor

*lv\_point\_t* **vertex\_p**

int32\_t **start\_angle**

int32\_t **end\_angle**

struct *lv\_draw\_sw\_mask\_angle\_param\_t* **cfg**

*lv\_draw\_sw\_mask\_line\_param\_t* **start\_line**

*lv\_draw\_sw\_mask\_line\_param\_t* **end\_line**

```
uint16_t delta_deg

struct _lv_draw_sw_mask_radius_param_t
```

**Public Members**

**lv\_draw\_sw\_mask\_common\_dsc\_t dsc**

The first element must be the common descriptor

*lv\_area\_t rect*

**int32\_t radius**

**uint8\_t outer**

Invert the mask. 0: Keep the pixels inside.

struct *\_lv\_draw\_sw\_mask\_radius\_param\_t* **cfg**

*lv\_draw\_sw\_mask\_radius\_circle\_dsc\_t \*circle*

```
struct _lv_draw_sw_mask_fade_param_t
```

**Public Members**

**lv\_draw\_sw\_mask\_common\_dsc\_t dsc**

The first element must be the common descriptor

*lv\_area\_t coords*

**int32\_t y\_top**

**int32\_t y\_bottom**

**lv\_opa\_t opa\_top**

**lv\_opa\_t opa\_bottom**

struct *\_lv\_draw\_sw\_mask\_fade\_param\_t* **cfg**

```
struct _lv_draw_sw_mask_map_param_t
```

**Public Members**

**lv\_draw\_sw\_mask\_common\_dsc\_t dsc**

The first element must be the common descriptor

***lv\_area\_t* coords**

```
const lv_opa_t *map

struct _lv_draw_sw_mask_map_param_t cfg
```

**lv\_draw\_sw\_private.h**

```
struct lv_draw_sw_thread_dsc_t
```

**Public Members**

```
lv_draw_task_t *task_act
```

```
lv_thread_t thread
```

```
lv_thread_sync_t sync
```

```
lv_draw_unit_t *draw_unit
```

```
uint32_t idx
```

```
volatile bool init
```

```
volatile bool exit_status
```

```
struct _lv_draw_sw_unit_t
```

**Public Members**

```
lv_draw_unit_t base_unit
```

```
lv_draw_task_t *task_act
```

**lv\_draw\_sw\_utils.h****Functions**

```
void lv_draw_sw_i1_to_argb8888(const void *buf_i1, void *buf_argb8888, uint32_t width, uint32_t
height, uint32_t buf_i1_stride, uint32_t buf_argb8888_stride,
uint32_t index0_color, uint32_t index1_color)
```

Converts an I1 buffer to ARGB8888 format.

**Parametri**

- **buf\_i1** -- pointer to buffer with I1 formatted render
- **buf\_argb8888** -- pointer to buffer for ARGB8888 render
- **width** -- width in pixels of the area. must be a multiple of 8.
- **height** -- height in pixels of the area
- **buf\_i1\_stride** -- stride of i1 buffer in bytes
- **buf\_argb8888\_stride** -- stride of argb8888 buffer in bytes
- **index0\_color** -- color of the 0 bits of i1 buf

- **index1\_color** -- color of the 1 bits of i1 buf

**void lv\_draw\_sw\_rgb565\_swap(void \*buf, uint32\_t buf\_size\_px)**

Swap the upper and lower byte of an RGB565 buffer. Might be required if a 8bit parallel port or an SPI port send the bytes in the wrong order. The bytes will be swapped in place.

**Parametri**

- **buf** -- pointer to buffer
- **buf\_size\_px** -- number of pixels in the buffer

**void lv\_draw\_sw\_i1\_invert(void \*buf, uint32\_t buf\_size)**

Invert a draw buffer in the I1 color format. Conventionally, a bit is set to 1 during blending if the luminance is greater than 127. Depending on the display controller used, you might want to have different behavior. The inversion will be performed in place.

**Parametri**

- **buf** -- pointer to the buffer to be inverted
- **buf\_size** -- size of the buffer in bytes

**void lv\_draw\_sw\_i1\_convert\_to\_vtiled(const void \*buf, uint32\_t buf\_size, uint32\_t width, uint32\_t height, void \*out\_buf, uint32\_t out\_buf\_size, bool bit\_order\_lsb)**

Convert a draw buffer in I1 color format from htiled (row-wise) to vtiled (column-wise) buffer layout. The conversion assumes that the buffer width and height is rounded to a multiple of 8.

**Parametri**

- **buf** -- pointer to the buffer to be converted
- **buf\_size** -- size of the buffer in bytes
- **width** -- width of the buffer
- **height** -- height of the buffer
- **out\_buf** -- pointer to the output buffer
- **out\_buf\_size** -- size of the output buffer in bytes
- **bit\_order\_lsb** -- bit order of the resulting vtiled buffer

**void lv\_draw\_sw\_rotate(const void \*src, void \*dest, int32\_t src\_width, int32\_t src\_height, int32\_t src\_stride, int32\_t dest\_stride, lv\_display\_rotation\_t rotation, lv\_color\_format\_t color\_format)**

Rotate a buffer into another buffer

**Parametri**

- **src** -- the source buffer
- **dest** -- the destination buffer
- **src\_width** -- source width in pixels
- **src\_height** -- source height in pixels
- **src\_stride** -- source stride in bytes (number of bytes in a row)
- **dest\_stride** -- destination stride in bytes (number of bytes in a row)
- **rotation** -- LV\_DISPLAY\_ROTATION\_0/90/180/270
- **color\_format** -- LV\_COLOR\_FORMAT\_RGB565/RGB888/XRGB8888/ARGB8888

**arm2d**

**lv\_draw\_sw\_arm2d.h**

**lv\_draw\_sw\_helium.h**

**blend**

**lv\_draw\_sw\_blend.h**

**Typedef**

**typedef void (\*lv\_draw\_sw\_blend\_handler\_t)(lv\_draw\_task\_t \*t, const lv\_draw\_sw\_blend\_dsc\_t \*dsc)**

Custom draw function for SW rendering.

**Param t**

pointer to a draw task

**Param dsc**

pointer to an initialized blend descriptor

**Functions**void **lv\_draw\_sw\_blend**(lv\_draw\_task\_t \*t, const lv\_draw\_sw\_blend\_dsc\_t \*dsc)

Call the blend function of the layer.

**Parametri**

- **draw\_unit** -- pointer to a draw unit
- **dsc** -- pointer to an initialized blend descriptor

struct **lv\_draw\_sw\_custom\_blend\_handler\_t****Public Members***lv\_color\_format\_t dest\_cf**lv\_draw\_sw\_blend\_handler\_t handler***lv\_draw\_sw\_blend\_private.h**struct **\_lv\_draw\_sw\_blend\_dsc\_t****Public Members**const *lv\_area\_t* \***blend\_area**The area with absolute coordinates to draw on `layer->buf` will be clipped to `layer->clip_area`const void \***src\_buf**Pointer to an image to blend. If set `fill_color` is ignored**uint32\_t src\_stride***lv\_color\_format\_t src\_color\_format*const *lv\_area\_t* \***src\_area****lv\_opa\_t opa**

The overall opacity

*lv\_color\_t color*

Fill color

const `lv_opa_t` \***mask\_buf**NULL if ignored, or an alpha mask to apply on `blend_area`*lv\_draw\_sw\_mask\_res\_t mask\_res*

The result of the previous mask operation

```
const lv_area_t *mask_area
```

The area of **mask\_buf** with absolute coordinates

```
int32_t mask_stride
```

```
lv_blend_mode_t blend_mode
```

E.g. LV\_BLEND\_MODE\_ADDITIVE

```
struct _lv_draw_sw_blend_fill_dsc_t
```

#### Public Members

```
void *dest_buf
```

```
int32_t dest_w
```

```
int32_t dest_h
```

```
int32_t dest_stride
```

```
const lv_opa_t *mask_buf
```

```
int32_t mask_stride
```

```
lv_color_t color
```

```
lv_opa_t opa
```

```
lv_area_t relative_area
```

```
struct _lv_draw_sw_blend_image_dsc_t
```

#### Public Members

```
void *dest_buf
```

```
int32_t dest_w
```

```
int32_t dest_h
```

```
int32_t dest_stride
```

```
const lv_opa_t *mask_buf
```

```
int32_t mask_stride
```

```
const void *src_buf
int32_t src_stride
lv_color_format_t src_color_format
lv_opa_t opa
lv_blend_mode_t blend_mode
```

***lv\_area\_t relative\_area***

The blend area relative to the layer's buffer area.

***lv\_area\_t src\_area***

The original src area.

**lv\_draw\_sw\_blend\_to\_al88.h****Functions**

```
void lv_draw_sw_blend_color_to_al88(lv_draw_sw_blend_fill_dsc_t *dsc)
void lv_draw_sw_blend_image_to_al88(lv_draw_sw_blend_image_dsc_t *dsc)
```

**lv\_draw\_sw\_blend\_to\_argb8888.h****Functions**

```
void lv_draw_sw_blend_color_to_argb8888(lv_draw_sw_blend_fill_dsc_t *dsc)
void lv_draw_sw_blend_image_to_argb8888(lv_draw_sw_blend_image_dsc_t *dsc)
```

**lv\_draw\_sw\_blend\_to\_argb8888\_premultiplied.h****Functions**

```
void lv_draw_sw_blend_color_to_argb8888_premultiplied(lv_draw_sw_blend_fill_dsc_t
                                                       *dsc)
void lv_draw_sw_blend_image_to_argb8888_premultiplied(lv_draw_sw_blend_image_dsc_t
                                                       *dsc)
```

**lv\_draw\_sw\_blend\_to\_i1.h****Functions**

```
void lv_draw_sw_blend_color_to_i1(lv_draw_sw_blend_fill_dsc_t *dsc)
void lv_draw_sw_blend_image_to_i1(lv_draw_sw_blend_image_dsc_t *dsc)
```

**lv\_draw\_sw\_blend\_to\_l8.h****Functions**

```
void lv_draw_sw_blend_color_to_l8(lv_draw_sw_blend_fill_dsc_t *dsc)
void lv_draw_sw_blend_image_to_l8(lv_draw_sw_blend_image_dsc_t *dsc)
```

**lv\_draw\_sw\_blend\_to\_rgb565.h****Functions**

```
void lv_draw_sw_blend_color_to_rgb565(lv_draw_sw_blend_fill_dsc_t *dsc)
```

```
void lv_draw_sw_blend_image_to_rgb565(lv_draw_sw_blend_image_dsc_t *dsc)
```

**lv\_draw\_sw\_blend\_to\_rgb565\_swapped.h****Functions**

```
void lv_draw_sw_blend_color_to_rgb565_swapped(lv_draw_sw_blend_fill_dsc_t *dsc)
```

```
void lv_draw_sw_blend_image_to_rgb565_swapped(lv_draw_sw_blend_image_dsc_t *dsc)
```

**lv\_draw\_sw\_blend\_to\_rgb888.h****Functions**

```
void lv_draw_sw_blend_color_to_rgb888(lv_draw_sw_blend_fill_dsc_t *dsc, uint32_t dest_px_size)
```

```
void lv_draw_sw_blend_image_to_rgb888(lv_draw_sw_blend_image_dsc_t *dsc, uint32_t dest_px_size)
```

**arm2d****lv\_blend\_arm2d.h****helium****lv\_blend\_helium.h****neon****lv\_blend\_neon.h****14.12.32 vg\_lite****lv\_draw\_vg\_lite.h****Functions**

```
void lv_draw_buf_vg_lite_init_handlers(void)
```

```
void lv_draw_vg_lite_init(void)
```

```
void lv_draw_vg_lite_deinit(void)
```

```
void lv_draw_vg_lite_arc(lv_draw_task_t *t, const lv_draw_arc_dsc_t *dsc, const lv_area_t *coords)
```

```
void lv_draw_vg_lite_box_shadow(lv_draw_task_t *t, const lv_draw_box_shadow_dsc_t *dsc, const lv_area_t *coords)
```

```
void lv_draw_vg_lite_border(lv_draw_task_t *t, const lv_draw_border_dsc_t *dsc, const lv_area_t *coords)
```

```
void lv_draw_vg_lite_fill(lv_draw_task_t *t, const lv_draw_fill_dsc_t *dsc, const lv_area_t *coords)
```

```
void lv_draw_vg_lite_img(lv_draw_task_t *t, const lv_draw_image_dsc_t *dsc, const lv_area_t *coords, bool no_cache)
```

```

void lv_draw_vg_lite_label_init(struct _lv_draw_vg_lite_unit_t *u)
void lv_draw_vg_lite_label_deinit(struct _lv_draw_vg_lite_unit_t *u)
void lv_draw_vg_lite_letter(lv_draw_task_t *t, const lv_draw_letter_dsc_t *dsc, const lv_area_t
                           *coords)
void lv_draw_vg_lite_label(lv_draw_task_t *t, const lv_draw_label_dsc_t *dsc, const lv_area_t
                           *coords)
void lv_draw_vg_lite_layer(lv_draw_task_t *t, const lv_draw_image_dsc_t *draw_dsc, const lv_area_t
                           *coords)
void lv_draw_vg_lite_line(lv_draw_task_t *t, const lv_draw_line_dsc_t *dsc)
void lv_draw_vg_lite_triangle(lv_draw_task_t *t, const lv_draw_triangle_dsc_t *dsc)
void lv_draw_vg_lite_mask_rect(lv_draw_task_t *t, const lv_draw_mask_rect_dsc_t *dsc, const
                               lv_area_t *coords)
void lv_draw_vg_lite_vector(lv_draw_task_t *t, const lv_draw_vector_task_dsc_t *dsc)

```

## lv\_draw\_vg\_lite\_type.h

struct \_lv\_draw\_vg\_lite\_unit\_t

### Public Members

lv\_draw\_unit\_t **base\_unit**

lv\_draw\_task\_t \***task\_act**

*lv\_area\_t* **current\_scissor\_area**

struct \_lv\_vg\_lite\_pending\_t \***image\_dsc\_pending**

struct \_lv\_vg\_lite\_grad\_ctx\_t \***grad\_ctx**

lv\_cache\_t \***stroke\_cache**

struct \_lv\_vg\_lite\_pending\_t \***bitmap\_font\_pending**

uint16\_t **flush\_count**

uint16\_t **letter\_count**

vg\_lite\_buffer\_t **target\_buffer**

vg\_lite\_matrix\_t **global\_matrix**

struct \_lv\_vg\_lite\_path\_t \***global\_path**

bool **path\_in\_use**

**lv\_vg\_lite\_decoder.h****Functions**

```
void lv_vg_lite_decoder_init(void)
void lv_vg_lite_decoder_deinit(void)
```

**lv\_vg\_lite\_grad.h****Functions**

```
struct _lv_vg_lite_grad_ctx_t *lv_vg_lite_grad_ctx_create(uint32_t cache_cnt, struct
                                                       _lv_draw_vg_lite_unit_t *unit)
```

Create a gradient context.

**Parametri**

- **cache\_cnt** -- number of cache entries
- **unit** -- the draw unit

```
void lv_vg_lite_grad_ctx_delete(struct _lv_vg_lite_grad_ctx_t *ctx)
```

Delete a gradient context.

**Parametri**

**ctx** -- the gradient context to delete

```
struct _lv_vg_lite_pending_t *lv_vg_lite_grad_ctx_get_pending(struct _lv_vg_lite_grad_ctx_t *ctx)
```

Get the pending list of gradient items.

**Parametri**

**ctx** -- the gradient context

```
bool lv_vg_lite_draw_grad(struct _lv_vg_lite_grad_ctx_t *ctx, vg_lite_buffer_t *buffer, vg_lite_path_t
                           *path, const lv_vector_gradient_t *grad, const vg_lite_matrix_t *grad_matrix,
                           const vg_lite_matrix_t *matrix, vg_lite_fill_t fill, vg_lite_blend_t blend)
```

Draw a gradient.

**Parametri**

- **ctx** -- the gradient context
- **buffer** -- the target buffer
- **path** -- the path to draw the gradient on
- **grad** -- the gradient descriptor
- **grad\_matrix** -- the gradient matrix
- **matrix** -- the matrix to apply to the gradient
- **fill** -- the fill rule
- **blend** -- the blend mode

**Ritorna**

true: success, false: failed

```
bool lv_vg_lite_draw_grad_helper(struct _lv_vg_lite_grad_ctx_t *ctx, vg_lite_buffer_t *buffer,
                                 vg_lite_path_t *path, const lv_area_t *area, const lv_grad_dsc_t
                                 *grad_dsc, const vg_lite_matrix_t *matrix, vg_lite_fill_t fill,
                                 vg_lite_blend_t blend)
```

Draw a gradient helper.

**Parametri**

- **ctx** -- the gradient context
- **buffer** -- the target buffer
- **path** -- the path to draw the gradient on
- **area** -- the area to draw the gradient on
- **grad\_dsc** -- the gradient descriptor
- **matrix** -- the matrix to apply to the gradient
- **fill** -- the fill rule
- **blend** -- the blend mode

**Ritorna**

true: success, false: failed

**lv\_vg\_lite\_math.h****Defines****MATH\_PI****MATH\_HALF\_PI****MATH\_TWO\_PI****DEG\_TO\_RAD****RAD\_TO\_DEG****MATH\_TANF(x)****MATH\_SINF(x)****MATH\_COSF(x)****MATH\_ASINF(x)****MATH\_ACOSF(x)****MATH\_FABSF(x)****MATH\_SQRTF(x)****MATH\_RADIANS(deg)****MATH\_DEGREES(rad)****Functions**static inline bool **math\_zero**(float a)static inline bool **math\_equal**(float a, float b)float **math\_fast\_inv\_sqrtf**(float number)**lv\_vg\_lite\_path.h****Defines****PATH\_COORD\_MAX**

It is found that thorvg cannot handle large coordinates well. When the coordinates are larger than 4096, the calculation of tvgSwRle module will overflow in 32-bit system. So we use FLT\_MAX and FLT\_MIN to write the mark to bounding\_box to tell vg\_lite\_tvg not to add clip path to the current path.

**PATH\_COORD\_MIN****LV\_VG\_LITE\_PATH\_SET\_OP\_CODE(PTR, TYPE, OP\_CODE)****LV\_VG\_LITE\_PATH\_GET\_OP\_CODE(PTR)**

**Typedef**

```
typedef struct _lv_vg_lite_path_t lv_vg_lite_path_t
```

```
typedef struct _lv_draw_vg_lite_unit_t lv_draw_vg_lite_unit_t
```

```
typedef void (*lv_vg_lite_path_iter_cb_t)(void *user_data, uint8_t op_code, const float *data,  
uint32_t len)
```

**Functions**

```
void lv_vg_lite_path_init(lv_draw_vg_lite_unit_t *unit)
```

```
void lv_vg_lite_path_deinit(lv_draw_vg_lite_unit_t *unit)
```

```
lv_vg_lite_path_t *lv_vg_lite_path_create(vg_lite_format_t data_format)
```

```
void lv_vg_lite_path_destroy(lv_vg_lite_path_t *path)
```

```
lv_vg_lite_path_t *lv_vg_lite_path_get(lv_draw_vg_lite_unit_t *unit, vg_lite_format_t data_format)
```

```
void lv_vg_lite_path_drop(lv_draw_vg_lite_unit_t *unit, lv_vg_lite_path_t *path)
```

```
void lv_vg_lite_path_reset(lv_vg_lite_path_t *path, vg_lite_format_t data_format)
```

```
void lv_vg_lite_path_set_bounding_box_area(lv_vg_lite_path_t *path, const lv_area_t *area)
```

```
void lv_vg_lite_path_set_bounding_box(lv_vg_lite_path_t *path, float min_x, float min_y, float  
max_x, float max_y)
```

```
void lv_vg_lite_path_get_bounding_box(lv_vg_lite_path_t *path, float *min_x, float *min_y, float  
*max_x, float *max_y)
```

```
bool lv_vg_lite_path_update_bounding_box(lv_vg_lite_path_t *path)
```

```
void lv_vg_lite_path_set_transform(lv_vg_lite_path_t *path, const vg_lite_matrix_t *matrix)
```

```
void lv_vg_lite_path_set_quality(lv_vg_lite_path_t *path, vg_lite_quality_t quality)
```

```
vg_lite_path_t *lv_vg_lite_path_get_path(lv_vg_lite_path_t *path)
```

```
void lv_vg_lite_path_reserve_space(lv_vg_lite_path_t *path, size_t len)
```

```
void lv_vg_lite_path_move_to(lv_vg_lite_path_t *path, float x, float y)
```

```
void lv_vg_lite_path_line_to(lv_vg_lite_path_t *path, float x, float y)
```

```
void lv_vg_lite_path_quad_to(lv_vg_lite_path_t *path, float cx, float cy, float x, float y)
```

```
void lv_vg_lite_path_cubic_to(lv_vg_lite_path_t *path, float cx1, float cy1, float cx2, float cy2, float x,  
float y)
```

```
void lv_vg_lite_path_close(lv_vg_lite_path_t *path)
```

```
void lv_vg_lite_path_end(lv_vg_lite_path_t *path)
```

```
void lv_vg_lite_path_append_rect(lv_vg_lite_path_t *path, float x, float y, float w, float h, float r)
```

```
void lv_vg_lite_path_append_circle(lv_vg_lite_path_t *path, float cx, float cy, float rx, float ry)
```

```
void lv_vg_lite_path_append_arc_right_angle(lv_vg_lite_path_t *path, float start_x, float start_y,
                                             float center_x, float center_y, float end_x, float
                                             end_y)

void lv_vg_lite_path_append_arc(lv_vg_lite_path_t *path, float cx, float cy, float radius, float
                                 start_angle, float sweep, bool pie)

void lv_vg_lite_path_append_path(lv_vg_lite_path_t *dest, const lv_vg_lite_path_t *src)

uint8_t lv_vg_lite_vlc_op_arg_len(uint8_t vlc_op)

uint8_t lv_vg_lite_path_format_len(vg_lite_format_t format)

void lv_vg_lite_path_for_each_data(const vg_lite_path_t *path, lv_vg_lite_path_iter_cb_t cb, void
                                   *user_data)
```

## lv\_vg\_lite\_pending.h

### Typedef

typedef struct \_lv\_vg\_lite\_pending\_t **lv\_vg\_lite\_pending\_t**

typedef void (\***lv\_vg\_lite\_pending\_free\_cb\_t**)(void \*obj, void \*user\_data)

### Functions

*lv\_vg\_lite\_pending\_t* \***lv\_vg\_lite\_pending\_create**(size\_t obj\_size, uint32\_t capacity\_default)

Create a pending list

#### Parametri

- **obj\_size** -- the size of the objects in the list
- **capacity\_default** -- the default capacity of the list

#### Ritorna

a pointer to the pending list

void **lv\_vg\_lite\_pending\_destroy**(*lv\_vg\_lite\_pending\_t* \*pending)

Destroy a pending list

#### Parametri

**pending** -- pointer to the pending list

void **lv\_vg\_lite\_pending\_set\_free\_cb**(*lv\_vg\_lite\_pending\_t* \*pending, *lv\_vg\_lite\_pending\_free\_cb\_t* free\_cb, void \*user\_data)

Set a free callback for the pending list

#### Parametri

- **pending** -- pointer to the pending list
- **free\_cb** -- the free callback
- **user\_data** -- user data to pass to the free callback

void **lv\_vg\_lite\_pending\_add**(*lv\_vg\_lite\_pending\_t* \*pending, void \*obj)

Add an object to the pending list

#### Parametri

- **pending** -- pointer to the pending list
- **obj** -- pointer to the object to add

void **lv\_vg\_lite\_pending\_remove\_all**(*lv\_vg\_lite\_pending\_t* \*pending)

Remove all objects from the active pending list

#### Parametri

**pending** -- pointer to the pending list

**void lv\_vg\_lite\_pending\_swap(*lv\_vg\_lite\_pending\_t* \*pending)**

Remove all old objects reference and swap new objects reference

**Parametri**

**pending** -- pointer to the pending list

## lv\_vg\_lite\_stroke.h

### Functions

**void lv\_vg\_lite\_stroke\_init(struct \_lv\_draw\_vg\_lite\_unit\_t \*unit, uint32\_t cache\_cnt)**

Initialize the stroke module

**Parametri**

**unit** -- pointer to the unit

**void lv\_vg\_lite\_stroke\_deinit(struct \_lv\_draw\_vg\_lite\_unit\_t \*unit)**

Deinitialize the stroke module

**Parametri**

**unit** -- pointer to the unit

**lv\_cache\_entry\_t \*lv\_vg\_lite\_stroke\_get(struct \_lv\_draw\_vg\_lite\_unit\_t \*unit, struct \_lv\_vg\_lite\_path\_t \*path, const lv\_vector\_stroke\_dsc\_t \*dsc)**

Get the stroke cache entry

**Parametri**

- **unit** -- pointer to the unit
- **path** -- pointer to the path
- **dsc** -- pointer to the stroke descriptor

**Ritorna**

pointer to the stroke cache entry

**struct \_lv\_vg\_lite\_path\_t \*lv\_vg\_lite\_stroke\_get\_path(lv\_cache\_entry\_t \*cache\_entry)**

Get the path of a stroke

**Parametri**

**cache\_entry** -- pointer to the stroke cache entry

**Ritorna**

pointer to the path

**void lv\_vg\_lite\_stroke\_drop(struct \_lv\_draw\_vg\_lite\_unit\_t \*unit, lv\_cache\_entry\_t \*cache\_entry)**

Drop the stroke cache entry

**Parametri**

- **unit** -- pointer to the unit
- **stroke** -- pointer to the stroke

## lv\_vg\_lite\_utils.h

### Defines

**LV\_VG\_LITE\_ASSERT(expr)**

**LV\_VG\_LITE\_CHECK\_ERROR(expr, error\_handler)**

**LV\_VG\_LITE\_ASSERT\_PATH(path)**

**LV\_VG\_LITE\_ASSERT\_SRC\_BUFFER(buffer)**

**LV\_VG\_LITE\_ASSERT\_DEST\_BUFFER(buffer)**

**LV\_VG\_LITE\_ASSERT\_MATRIX(matrix)**

**LV\_VG\_LITE\_ALIGN(number, align\_bytes)**

**LV\_VG\_LITE\_IS\_ALIGNED(num, align)**

**LV\_VG\_LITE\_IS\_INDEX\_FMT(fmt)**

## Functions

```

void lv_vg_lite_dump_info(void)

void lv_vg_lite_error_dump_info(vg_lite_error_t error)

const char *lv_vg_lite_error_string(vg_lite_error_t error)

const char *lv_vg_lite_feature_string(vg_lite_feature_t feature)

const char *lv_vg_lite_buffer_format_string(vg_lite_buffer_format_t format)

const char *lv_vg_lite_vlc_op_string(uint8_t vlc_op)

void lv_vg_lite_path_dump_info(const vg_lite_path_t *path)

void lv_vg_lite_stroke_dump_info(const vg_lite_stroke_t *stroke)

void lv_vg_lite_buffer_dump_info(const vg_lite_buffer_t *buffer)

void lv_vg_lite_matrix_dump_info(const vg_lite_matrix_t *matrix)

void lv_vg_lite_color_dump_info(const vg_lite_color_t color)

bool lv_vg_lite_is_dest_cf_supported(lv_color_format_t cf)

bool lv_vg_lite_is_src_cf_supported(lv_color_format_t cf)

vg_lite_buffer_format_t lv_vg_lite_vg_fmt(lv_color_format_t cf)

void lv_vg_lite_buffer_format_bytes(vg_lite_buffer_format_t format, uint32_t *mul, uint32_t *div,
                                    uint32_t *bytes_align)

uint32_t lv_vg_lite_width_to_stride(uint32_t w, vg_lite_buffer_format_t color_format)

void lv_vg_lite_buffer_init(vg_lite_buffer_t *buffer, const void *ptr, int32_t width, int32_t height,
                           uint32_t stride, vg_lite_buffer_format_t format, bool tiled)

void lv_vg_lite_buffer_from_draw_buf(vg_lite_buffer_t *buffer, const lv_draw_buf_t *draw_buf)

void lv_vg_lite_image_matrix(vg_lite_matrix_t *matrix, int32_t x, int32_t y, const
                            lv_draw_image_dsc_t *dsc)

vg_lite_color_t lv_vg_lite_image_recolor(vg_lite_buffer_t *buffer, const lv_draw_image_dsc_t *dsc)

bool lv_vg_lite_buffer_open_image(vg_lite_buffer_t *buffer, lv_image_decoder_dsc_t *decoder_dsc,
                                 const void *src, bool no_cache, bool premultiply)

void lv_vg_lite_image_dsc_init(struct lv_draw_vg_lite_unit_t *unit)

void lv_vg_lite_image_dsc_deinit(struct lv_draw_vg_lite_unit_t *unit)

vg_lite_blend_t lv_vg_lite_blend_mode(lv_blend_mode_t blend_mode, bool has_pre_mul)

uint32_t lv_vg_lite_get_palette_size(vg_lite_buffer_format_t format)

vg_lite_color_t lv_vg_lite_color(lv_color_t color, lv_opa_t opa, bool pre_mul)

void lv_vg_lite_rect(vg_lite_rectangle_t *rect, const lv_area_t *area)

static inline void lv_vg_lite_matrix(vg_lite_matrix_t *dest, const lv_matrix_t *src)

bool lv_vg_lite_buffer_check(const vg_lite_buffer_t *buffer, bool is_src)

bool lv_vg_lite_path_check(const vg_lite_path_t *path)

```

```

bool lv_vg_lite_matrix_check(const vg_lite_matrix_t *matrix)
bool lv_vg_lite_support_blend_normal(void)
void lv_vg_lite_matrix_multiply(vg_lite_matrix_t *matrix, const vg_lite_matrix_t *mult)
bool lv_vg_lite_matrix_inverse(vg_lite_matrix_t *result, const vg_lite_matrix_t *matrix)
lv_point_precise_t lv_vg_lite_matrix_transform_point(const vg_lite_matrix_t *matrix, const
                                                      lv_point_precise_t *point)

void lv_vg_lite_set_scissor_area(struct _lv_draw_vg_lite_unit_t *u, const lv_area_t *area)
void lv_vg_lite_disable_scissor(void)
void lv_vg_lite_flush(struct _lv_draw_vg_lite_unit_t *u)
void lv_vg_lite_finish(struct _lv_draw_vg_lite_unit_t *u)

static inline void lv_vg_lite_draw(vg_lite_buffer_t *target, vg_lite_path_t *path, vg_lite_fill_t fill_rule,
                                   vg_lite_matrix_t *matrix, vg_lite_blend_t blend, vg_lite_color_t color)

static inline void lv_vg_lite_draw_pattern(vg_lite_buffer_t *target, vg_lite_path_t *path, vg_lite_fill_t
                                         fill_rule, vg_lite_matrix_t *path_matrix, vg_lite_buffer_t
                                         *pattern_image, vg_lite_matrix_t *pattern_matrix,
                                         vg_lite_blend_t blend, vg_lite_pattern_mode_t
                                         pattern_mode, vg_lite_color_t pattern_color, vg_lite_color_t
                                         color, vg_lite_filter_t filter)

static inline void lv_vg_lite.blit_rect(vg_lite_buffer_t *target, vg_lite_buffer_t *source,
                                       vg_lite_rectangle_t *rect, vg_lite_matrix_t *matrix,
                                       vg_lite_blend_t blend, vg_lite_color_t color, vg_lite_filter_t
                                       filter)

static inline void lv_vg_lite_clear(vg_lite_buffer_t *target, const lv_area_t *area, vg_lite_color_t color)

```

## 14.13 drivers

### 14.13.1 lv\_drivers.h

#### 14.13.2 display

drm

[lv\\_linux\\_drm.h](#)

##### Functions

```

lv_display_t *lv_linux_drm_create(void)
void lv_linux_drm_set_file(lv_display_t *disp, const char *file, int64_t connector_id)

```

fb

[lv\\_linux\\_fbdev.h](#)

##### Functions

```

lv_display_t *lv_linux_fbdev_create(void)
void lv_linux_fbdev_set_file(lv_display_t *disp, const char *file)

```

```
void lv_linux_fbdev_set_force_refresh(lv_display_t *disp, bool enabled)
    Force the display to be refreshed on every change. Expected to be used with LV_DISPLAY_RENDER_MODE_DIRECT or LV_DISPLAY_RENDER_MODE_FULL.
```

**ft81x****lv\_ft81x.h****Typedef**

```
typedef void (*lv_ft81x_spi_cb_t)(lv_display_t *disp, lv_ft81x_spi_operation_t operation, void *data, uint32_t length)
```

**Enums****enum lv\_ft81x\_spi\_operation\_t**

*Values:*

enumerator **LV\_FT81X\_SPI\_OPERATION\_CS\_ASSERT**

enumerator **LV\_FT81X\_SPI\_OPERATION\_CS\_DEASSERT**

enumerator **LV\_FT81X\_SPI\_OPERATION\_SEND**

enumerator **LV\_FT81X\_SPI\_OPERATION\_RECEIVE**

**Functions**

```
lv_display_t *lv_ft81x_create(const lv_ft81x_parameters_t *params, void *partial_buf, uint32_t buf_size,
                               lv_ft81x_spi_cb_t spi_cb, void *user_data)
```

Create a framebuffer-based ft81x driver display.

**Parametri**

- **params** -- pointer to a struct of display panel properties. does not need to be static.
- **partial\_buf** -- a single partial buffer
- **buf\_size** -- size of the partial buffer
- **spi\_cb** -- a callback called by the driver to perform SPI operations
- **user\_data** -- use **lv\_ft81x\_get\_user\_data** to get this pointer inside the SPI callback

**Ritorna**

pointer to the display

```
void *lv_ft81x_get_user_data(lv_display_t *disp)
```

Get the **user\_data** parameter that was passed to **lv\_ft81x\_create**. Useful in the SPI callback.

**Parametri**

**disp** -- pointer to the ft81x display

**Ritorna**

the **user\_data** pointer

```
struct lv_ft81x_parameters_t
```

**Public Members**

**uint16\_t hor\_res**  
active display width

**uint16\_t ver\_res**  
active display height

**uint16\_t hcycle**  
total number of clocks per line, incl front/back porch

**uint16\_t hoffset**  
start of active line

**uint16\_t hsync0**  
start of horizontal sync pulse

**uint16\_t hsync1**  
end of horizontal sync pulse

**uint16\_t vcycle**  
total number of lines per screen, including pre/post

**uint16\_t voffset**  
start of active screen

**uint16\_t vsync0**  
start of vertical sync pulse

**uint16\_t vsync1**  
end of vertical sync pulse

**uint8\_t swizzle**  
FT8xx output to LCD - pin order

**uint8\_t pclkpol**  
LCD data is clocked in on this PCLK edge

**uint8\_t cspread**  
helps with noise, when set to 1 fewer signals are changed simultaneously, reset-default: 1

**uint8\_t pclk**  
60MHz / pclk = pclk frequency

**bool has\_crystal**  
has an external clock crystal

**bool is\_bt81x**  
is a BT series model, not FT

**lv\_ft81x\_defines.h**

Copied from [https://github.com/lvgl/lvgl\\_esp32\\_drivers/blob/9fed1cc47b5a45fec6bae08b55d2147d3b50260c/lvgl\\_tft/EVE.h](https://github.com/lvgl/lvgl_esp32_drivers/blob/9fed1cc47b5a45fec6bae08b55d2147d3b50260c/lvgl_tft/EVE.h)

**Defines****FT81X\_ENABLE****DL\_CLEAR****DL\_CLEAR\_RGB****DL\_COLOR\_RGB****DL\_POINT\_SIZE****DL\_END****DL\_BEGIN****DL\_DISPLAY****CLR\_COL****CLR\_STN****CLR\_TAG****SPI\_WIDTH\_SIO****SPI\_WIDTH\_DIO****SPI\_WIDTH\_QIO****EVE\_ACTIVE****EVE\_STANDBY****EVE\_SLEEP****EVE\_PWRDOWN****EVE\_CLKEXT****EVE\_CLKINT**

EVE\_CORERST

EVE\_CLK48M

EVE\_CLK36M

EVE\_NEVER

EVE\_LESS

EVE\_EQUAL

EVE\_GREATER

EVE\_GEQUAL

EVE\_EQUAL

EVE\_NOTEQUAL

EVE\_ALWAYS

EVE\_ARGB1555

EVE\_L1

EVE\_L4

EVE\_L8

EVE\_RGB332

EVE\_ARGB2

EVE\_ARGB4

EVE\_RGB565

EVE\_PALETTED

EVE\_TEXT8X8

EVE\_TEXTVGA

EVE\_BARGRAPH

**EVE\_NEAREST**

**EVE\_BILINEAR**

**EVE\_BORDER**

**EVE\_REPEAT**

**EVE\_KEEP**

**EVE\_REPLACE**

**EVE\_INCR**

**EVE\_DECR**

**EVE\_INVERT**

**EVE\_DLSWAP\_DONE**

**EVE\_DLSWAP\_LINE**

**EVE\_DLSWAP\_FRAME**

**EVE\_INT\_SWAP**

**EVE\_INT\_TOUCH**

**EVE\_INT\_TAG**

**EVE\_INT\_SOUND**

**EVE\_INT\_PLAYBACK**

**EVE\_INT\_CMDEMPTY**

**EVE\_INT\_CMDFLAG**

**EVE\_INT\_CONVCOMPLETE**

**EVE\_TMODE\_OFF**

**EVE\_TMODE\_ONESHOT**

**EVE\_TMODE\_FRAME**

**EVE\_TMODE\_CONTINUOUS**

**EVE\_ZERO**

**EVE\_ONE**

**EVE\_SRC\_ALPHA**

**EVE\_DST\_ALPHA**

**EVE\_ONE\_MINUS\_SRC\_ALPHA**

**EVE\_ONE\_MINUS\_DST\_ALPHA**

**EVE\_BITMAPS**

**EVE\_POINTS**

**EVE\_LINES**

**EVE\_LINE\_STRIP**

**EVE\_EDGE\_STRIP\_R**

**EVE\_EDGE\_STRIP\_L**

**EVE\_EDGE\_STRIP\_A**

**EVE\_EDGE\_STRIP\_B**

**EVE\_RECTS**

**EVE\_OPT\_MONO**

**EVE\_OPT\_NODL**

**EVE\_OPT\_FLAT**

**EVE\_OPT\_CENTERX**

**EVE\_OPT\_CENTERY**

**EVE\_OPT\_CENTER**

**EVE\_OPT\_NOBACK**

**EVE\_OPT\_NOTICKS**

**EVE\_OPT\_NOHM**

**EVE\_OPT\_NOPINTER**

**EVE\_OPT\_NOSECS**

**EVE\_OPT\_NOHANDS**

**EVE\_OPT\_RIGHTX**

**EVE\_OPT\_SIGNED**

**EVE\_NUMCHAR\_PERFONT**

**EVE\_FONT\_TABLE\_SIZE**

**EVE\_FONT\_TABLE\_POINTER**

**EVE\_LINEAR\_SAMPLES**

**EVE\_ULAW\_SAMPLES**

**EVE\_ADPCM\_SAMPLES**

**EVE\_SILENCE**

**EVE\_SQUAREWAVE**

**EVE\_SINEWAVE**

**EVE\_SAWTOOTH**

**EVE\_TRIANGLE**

**EVE\_BEEPING**

**EVE\_ALARM**

**EVE\_WARBLE**

**EVE\_CAROUSEL**

**EVE\_PIPS(n)**

**EVE\_HARP**

**EVE\_XYLOPHONE**

**EVE\_TUBA**

**EVE\_GLOCKENSPIEL**

**EVE\_ORGAN**

**EVE\_TRUMPET**

**EVE\_PIANO**

**EVE\_CHIMES**

**EVE\_MUSICBOX**

**EVE\_BELL**

**EVE\_CLICK**

**EVE\_SWITCH**

**EVE\_COWBELL**

**EVE\_NOTCH**

**EVE\_HIHAT**

**EVE\_KICKDRUM**

**EVE\_POP**

**EVE\_CLACK**

**EVE\_CHACK**

**EVE\_MUTE**

**EVE\_UNMUTE**

**EVE\_MIDI\_A0**

**EVE\_MIDI\_A\_0**

EVE\_MIDI\_B0

EVE\_MIDI\_C1

EVE\_MIDI\_C\_1

EVE\_MIDI\_D1

EVE\_MIDI\_D\_1

EVE\_MIDI\_E1

EVE\_MIDI\_F1

EVE\_MIDI\_F\_1

EVE\_MIDI\_G1

EVE\_MIDI\_G\_1

EVE\_MIDI\_A1

EVE\_MIDI\_A\_1

EVE\_MIDI\_B1

EVE\_MIDI\_C2

EVE\_MIDI\_C\_2

EVE\_MIDI\_D2

EVE\_MIDI\_D\_2

EVE\_MIDI\_E2

EVE\_MIDI\_F2

EVE\_MIDI\_F\_2

EVE\_MIDI\_G2

EVE\_MIDI\_G\_2

EVE\_MIDI\_A2

EVE\_MIDI\_A\_2

EVE\_MIDI\_B2

EVE\_MIDI\_C3

EVE\_MIDI\_C\_3

EVE\_MIDI\_D3

EVE\_MIDI\_D\_3

EVE\_MIDI\_E3

EVE\_MIDI\_F3

EVE\_MIDI\_F\_3

EVE\_MIDI\_G3

EVE\_MIDI\_G\_3

EVE\_MIDI\_A3

EVE\_MIDI\_A\_3

EVE\_MIDI\_B3

EVE\_MIDI\_C4

EVE\_MIDI\_C\_4

EVE\_MIDI\_D4

EVE\_MIDI\_D\_4

EVE\_MIDI\_E4

EVE\_MIDI\_F4

EVE\_MIDI\_F\_4

EVE\_MIDI\_G4

EVE\_MIDI\_G\_4

EVE\_MIDI\_A4

EVE\_MIDI\_A\_4

EVE\_MIDI\_B4

EVE\_MIDI\_C5

EVE\_MIDI\_C\_5

EVE\_MIDI\_D5

EVE\_MIDI\_D\_5

EVE\_MIDI\_E5

EVE\_MIDI\_F5

EVE\_MIDI\_F\_5

EVE\_MIDI\_G5

EVE\_MIDI\_G\_5

EVE\_MIDI\_A5

EVE\_MIDI\_A\_5

EVE\_MIDI\_B5

EVE\_MIDI\_C6

EVE\_MIDI\_C\_6

EVE\_MIDI\_D6

EVE\_MIDI\_D\_6

EVE\_MIDI\_E6

EVE\_MIDI\_F6

EVE\_MIDI\_F\_6

EVE\_MIDI\_G6

EVE\_MIDI\_G\_6

EVE\_MIDI\_A6

EVE\_MIDI\_A\_6

EVE\_MIDI\_B6

EVE\_MIDI\_C7

EVE\_MIDI\_C\_7

EVE\_MIDI\_D7

EVE\_MIDI\_D\_7

EVE\_MIDI\_E7

EVE\_MIDI\_F7

EVE\_MIDI\_F\_7

EVE\_MIDI\_G7

EVE\_MIDI\_G\_7

EVE\_MIDI\_A7

EVE\_MIDI\_A\_7

EVE\_MIDI\_B7

EVE\_MIDI\_C8

EVE\_GPIO00

EVE\_GPIO1

EVE\_GPIO7

EVE\_DISPLAY\_0

EVE\_DISPLAY\_180

CMD\_APPEND

CMD\_BGCOLOR

CMD\_BUTTON

CMD\_CALIBRATE

CMD\_CLOCK

CMD\_COLDSTART

CMD\_DIAL

CMD\_DLSTART

CMD\_FGCOLOR

CMD\_GAUGE

CMD\_GETMATRIX

CMD\_GETPROPS

CMD\_GETPTR

CMD\_GRADCOLOR

CMD\_GRADIENT

CMD\_INFLATE

CMD\_INTERRUPT

CMD\_KEYS

CMD\_LOADIDENTITY

CMD\_LOADIMAGE

CMD\_LOGO

CMD\_MEMCPY

CMD\_MEMCRC

CMD\_MEMSET

**CMD\_MEMWRITE**

**CMD\_MEMZERO**

**CMD\_NUMBER**

**CMD\_PROGRESS**

**CMD\_REGREAD**

**CMD\_ROTATE**

**CMD\_SCALE**

**CMD\_SCREENSAVER**

**CMD\_SCROLLBAR**

**CMD\_SETFONT**

**CMD\_SETMATRIX**

**CMD\_SKETCH**

**CMD\_SLIDER**

**CMD\_SNAPSHOT**

**CMD\_SPINNER**

**CMD\_STOP**

**CMD\_SWAP**

**CMD\_TEXT**

**CMD\_TOGGLE**

**CMD\_TRACK**

**CMD\_TRANSLATE**

**ALPHA\_FUNC(func, ref)**

**BEGIN(prim)**

**BITMAP\_HANDLE(handle)**

**BITMAP\_LAYOUT**(format, linestride, height)  
**BITMAP\_SIZE**(filter, wrapx, wrapy, width, height)  
**BITMAP\_TRANSFORM\_A**(a)  
**BITMAP\_TRANSFORM\_B**(b)  
**BITMAP\_TRANSFORM\_C**(c)  
**BITMAP\_TRANSFORM\_D**(d)  
**BITMAP\_TRANSFORM\_E**(e)  
**BITMAP\_TRANSFORM\_F**(f)  
**BLEND\_FUNC**(src, dst)  
**CALL**(dest)  
**CELL**(cell)  
**CLEAR**(c, s, t)  
**CLEAR\_COLOR\_A**(alpha)  
**CLEAR\_COLOR\_RGB**(red, green, blue)  
**CLEAR\_STENCIL**(s)  
**CLEAR\_TAG**(s)  
**COLOR\_A**(alpha)  
**COLOR\_MASK**(r, g, b, a)  
**COLOR\_RGB**(red, green, blue)  
**END()**  
**JUMP**(dest)  
**LINE\_WIDTH**(width)  
**MACRO**(m)  
**POINT\_SIZE**(size)  
**RESTORE\_CONTEXT()**  
**RETURN()**  
**SAVE\_CONTEXT()**  
**STENCIL\_FUNC**(func, ref, mask)  
**STENCIL\_MASK**(mask)  
**STENCIL\_OP**(sfail, spass)  
**TAG**(s)  
**TAG\_MASK**(mask)  
**VERTEX2F**(x, y)

**VERTEX2II**(x, y, handle, cell)

**EVE\_CLKSEL**

**EVE\_RST\_PULSE**

**EVE\_PINDRIVE**

**EVE\_PIN\_PD\_STATE**

**EVE\_RAM\_G**

**EVE\_ROM\_CHIPID**

**EVE\_ROM\_FONT**

**EVE\_ROM\_FONT\_ADDR**

**EVE\_RAM\_DL**

**EVE\_RAM\_REG**

**EVE\_RAM\_CMD**

**EVE\_RAM\_G\_SIZE**

**EVE\_CMDFIFO\_SIZE**

**EVE\_RAM\_DL\_SIZE**

**EVE\_ADC\_DIFFERENTIAL**

**EVE\_ADC\_SINGLE\_ENDED**

**EVE\_INT\_G8**

**EVE\_INT\_L8C**

**EVE\_INT\_VGA**

**EVE\_OPT\_MEDIAFIFO**

**EVE\_OPT\_FULLSCREEN**

**EVE\_OPT\_NOTEAR**

**EVE\_OPT\_SOUND**

**EVE\_PALETTED565**

**EVE\_PALETTED4444**

**EVE\_PALETTED8**

**EVE\_L2**

**CMD\_MEDIAFIFO**

**CMD\_PLAYVIDEO**

**CMD\_ROMFONT**

**CMD\_SETBASE**

**CMD\_SETBITMAP**

**CMD\_SETFONT2**

**CMD\_SETROTATE**

**CMD\_SETSCRATCH**

**CMD\_SNAPSHOT2**

**CMD\_VIDEOFRAME**

**CMD\_VIDEOSTART**

**REG\_ANA\_COMP**

**REG\_BIST\_EN**

**REG\_CLOCK**

**REG\_CMDB\_SPACE**

**REG\_CMDB\_WRITE**

**REG\_CMD\_DL**

**REG\_CMD\_READ**

**REG\_CMD\_WRITE**

**REG\_CPURESET**

**REG\_CSPREAD**

**REG\_CTOUCH\_EXTENDED**

**REG\_CTOUCH\_TOUCH0\_XY**

**REG\_CTOUCH\_TOUCH4\_X**

**REG\_CTOUCH\_TOUCH4\_Y**

**REG\_CTOUCH\_TOUCH1\_XY**

**REG\_CTOUCH\_TOUCH2\_XY**

**REG\_CTOUCH\_TOUCH3\_XY**

**REG\_TOUCH\_CONFIG**

**REG\_DATESTAMP**

**REG\_DITHER**

**REG\_DL\_SWAP**

**REG\_FRAMES**

**REG\_FREQUENCY**

**REG\_GPIO**

**REG\_GPIOX**

**REG\_GPIOX\_DIR**

**REG\_GPIO\_DIR**

**REG\_HCYCLE**

**REG\_HOFFSET**

**REG\_HSIZE**

`REG_HSYNC0`

`REG_HSYNC1`

`REG_ID`

`REG_INT_EN`

`REG_INT_FLAGS`

`REG_INT_MASK`

`REG_MACRO_0`

`REG_MACRO_1`

`REG_MEDIAFIFO_READ`

`REG_MEDIAFIFO_WRITE`

`REG_OUTBITS`

`REG_PCLK`

`REG_PCLK_POL`

`REG_PLAY`

`REG_PLAYBACK_FORMAT`

`REG_PLAYBACK_FREQ`

`REG_PLAYBACK_LENGTH`

`REG_PLAYBACK_LOOP`

`REG_PLAYBACK_PLAY`

`REG_PLAYBACK_READPTR`

`REG_PLAYBACK_START`

`REG_PWM_DUTY`

`REG_PWM_HZ`

**REG\_RENDERMODE**

**REG\_ROTATE**

**REG\_SNAPFORMAT**

**REG\_SNAPSHOT**

**REG\_SNAPY**

**REG\_SOUND**

**REG\_SPI\_WIDTH**

**REG\_SWIZZLE**

**REG\_TAG**

**REG\_TAG\_X**

**REG\_TAG\_Y**

**REG\_TAP\_CRC**

**REG\_TAP\_MASK**

**REG\_TOUCH\_ADC\_MODE**

**REG\_TOUCH\_CHARGE**

**REG\_TOUCH\_DIRECT\_XY**

**REG\_TOUCH\_DIRECT\_Z1Z2**

**REG\_TOUCH\_MODE**

**REG\_TOUCH\_OVERSAMPLE**

**REG\_TOUCH\_RAW\_XY**

**REG\_TOUCH\_RZ**

**REG\_TOUCH\_RZTHRESH**

**REG\_TOUCH\_SCREEN\_XY**

`REG_TOUCH_SETTLE`

`REG_TOUCH_TAG`

`REG_TOUCH_TAG1`

`REG_TOUCH_TAG1_XY`

`REG_TOUCH_TAG2`

`REG_TOUCH_TAG2_XY`

`REG_TOUCH_TAG3`

`REG_TOUCH_TAG3_XY`

`REG_TOUCH_TAG4`

`REG_TOUCH_TAG4_XY`

`REG_TOUCH_TAG_XY`

`REG_TOUCH_TRANSFORM_A`

`REG_TOUCH_TRANSFORM_B`

`REG_TOUCH_TRANSFORM_C`

`REG_TOUCH_TRANSFORM_D`

`REG_TOUCH_TRANSFORM_E`

`REG_TOUCH_TRANSFORM_F`

`REG_TRACKER`

`REG_TRACKER_1`

`REG_TRACKER_2`

`REG_TRACKER_3`

`REG_TRACKER_4`

`REG_TRIM`

**REG\_VCYCLE****REG\_VOFFSET****REG\_VOL\_PB****REG\_VOL\_SOUND****REG\_VSIZE****REG\_VSYNC0****REG\_VSYNC1****BITMAP\_LAYOUT\_H**(linestride, height)**BITMAP\_SIZE\_H**(width, height)**BITMAP\_SOURCE**(addr)**PALETTE\_SOURCE**(addr)**SCISSOR\_SIZE**(width, height)**SCISSOR\_XY**(x, y)**VERTEX\_FORMAT**(frac)**VERTEX\_TRANSLATE\_X**(x)**VERTEX\_TRANSLATE\_Y**(y)**ili9341****lv\_ili9341.h**

This driver is just a wrapper around the generic MIPI compatible LCD controller driver

## Typedef

```
typedef lv_lcd_send_cmd_cb_t lv_ili9341_send_cmd_cb_t
```

```
typedef lv_lcd_send_color_cb_t lv_ili9341_send_color_cb_t
```

## Functions

```
lv_display_t *lv_ili9341_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                                lv_ili9341_send_cmd_cb_t send_cmd_cb, lv_ili9341_send_color_cb_t
                                send_color_cb)
```

Create an LCD display with ILI9341 driver

### Parametri

- **hor\_res** -- horizontal resolution
- **ver\_res** -- vertical resolution
- **flags** -- default configuration settings (mirror, RGB ordering, etc.)
- **send\_cmd** -- platform-dependent function to send a command to the LCD controller (usually uses polling transfer)

- **send\_color** -- platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' callback)

**Ritorna**

pointer to the created display

**void lv\_il9341\_set\_gap(lv\_display\_t \*disp, uint16\_t x, uint16\_t y)**

Set gap, i.e., the offset of the (0,0) pixel in the VRAM

**Parametri**

- **disp** -- display object
- **x** -- x offset
- **y** -- y offset

**void lv\_il9341\_set\_invert(lv\_display\_t \*disp, bool invert)**

Set color inversion

**Parametri**

- **disp** -- display object
- **invert** -- false: normal, true: invert

**void lv\_il9341\_set\_gamma\_curve(lv\_display\_t \*disp, uint8\_t gamma)**

Set gamma curve

**Parametri**

- **disp** -- display object
- **gamma** -- gamma curve

**void lv\_il9341\_send\_cmd\_list(lv\_display\_t \*disp, const uint8\_t \*cmd\_list)**

Send list of commands.

**Parametri**

- **disp** -- display object
- **cmd\_list** -- controller and panel-specific commands

**lcd****lv\_lcd\_generic\_mipi.h**

Generic driver for controllers adhering to the MIPI DBI/DCS specification

Works with:

ST7735 ST7789 ST7796 ILI9341 ILI9488 (NOTE: in SPI mode ILI9488 only supports RGB666 mode, which is currently not supported)

any probably many more

**Defines**

**LV\_LCD\_CMD\_NOP**

**LV\_LCD\_CMD\_SOFT\_RESET**

**LV\_LCD\_CMD\_GET\_POWER\_MODE**

**LV\_LCD\_CMD\_GET\_ADDRESS\_MODE**

**LV\_LCD\_CMD\_GET\_PIXEL\_FORMAT**

**LV\_LCD\_CMD\_GET\_DISPLAY\_MODE**

`LV_LCD_CMD_GET_SIGNAL_MODE`  
`LV_LCD_CMD_GET_DIAGNOSTIC_RESULT`  
`LV_LCD_CMD_ENTER_SLEEP_MODE`  
`LV_LCD_CMD_EXIT_SLEEP_MODE`  
`LV_LCD_CMD_ENTER_PARTIAL_MODE`  
`LV_LCD_CMD_ENTER_NORMAL_MODE`  
`LV_LCD_CMD_EXIT_INVERT_MODE`  
`LV_LCD_CMD_ENTER_INVERT_MODE`  
`LV_LCD_CMD_SET_GAMMA_CURVE`  
`LV_LCD_CMD_SET_DISPLAY_OFF`  
`LV_LCD_CMD_SET_DISPLAY_ON`  
`LV_LCD_CMD_SET_COLUMN_ADDRESS`  
`LV_LCD_CMD_SET_PAGE_ADDRESS`  
`LV_LCD_CMD_WRITE_MEMORY_START`  
`LV_LCD_CMD_READ_MEMORY_START`  
`LV_LCD_CMD_SET_PARTIAL_ROWS`  
`LV_LCD_CMD_SET_PARTIAL_COLUMNS`  
`LV_LCD_CMD_SET_SCROLL_AREA`  
`LV_LCD_CMD_SET_TEAR_OFF`  
`LV_LCD_CMD_SET_TEAR_ON`  
`LV_LCD_CMD_SET_ADDRESS_MODE`  
`LV_LCD_CMD_SET_SCROLL_START`  
`LV_LCD_CMD_EXIT_IDLE_MODE`

LV\_LCD\_CMD\_ENTER\_IDLE\_MODE  
LV\_LCD\_CMD\_SET\_PIXEL\_FORMAT  
LV\_LCD\_CMD\_WRITE\_MEMORY\_CONTINUE  
LV\_LCD\_CMD\_READ\_MEMORY\_CONTINUE  
LV\_LCD\_CMD\_SET\_TEAR\_SCANLINE  
LV\_LCD\_CMD\_GET\_SCANLINE  
LV\_LCD\_CMD\_READ\_DDB\_CONTINUE  
LV\_LCD\_CMD\_READ\_DDB\_START  
LV\_LCD\_MASK\_FLIP\_VERTICAL  
LV\_LCD\_MASK\_FLIP\_HORIZONTAL  
LV\_LCD\_MASK\_DATA\_LATCH\_DATA\_ORDER  
LV\_LCD\_MASK\_RGB\_ORDER  
LV\_LCD\_MASK\_LINE\_ADDRESS\_ORDER  
LV\_LCD\_MASK\_PAGE\_COLUMN\_ORDER  
LV\_LCD\_MASK\_COLUMN\_ADDRESS\_ORDER  
LV\_LCD\_MASK\_PAGE\_ADDRESS\_ORDER  
LV\_LCD\_BIT\_FLIP\_VERTICAL\_\_NOT\_FLIPPED  
LV\_LCD\_BIT\_FLIP\_VERTICAL\_\_FLIPPED  
LV\_LCD\_BIT\_FLIP\_HORIZONTAL\_\_NOT\_FLIPPED  
LV\_LCD\_BIT\_FLIP\_HORIZONTAL\_\_FLIPPED  
LV\_LCD\_BIT\_DATA\_LATCH\_DATA\_ORDER\_\_LTOR  
LV\_LCD\_BIT\_DATA\_LATCH\_DATA\_ORDER\_\_RTOL  
LV\_LCD\_BIT\_RGB\_ORDER\_\_RGB

`LV_LCD_BIT_RGB_ORDER_BGR`

`LV_LCD_BIT_LINE_ADDRESS_ORDER_TT0B`

`LV_LCD_BIT_LINE_ADDRESS_ORDER_BT0T`

`LV_LCD_BIT_PAGE_COLUMN_ORDER_NORMAL`

`LV_LCD_BIT_PAGE_COLUMN_ORDER_REVERSE`

`LV_LCD_BIT_COLUMN_ADDRESS_ORDER_LT0R`

`LV_LCD_BIT_COLUMN_ADDRESS_ORDER_RT0L`

`LV_LCD_BIT_PAGE_ADDRESS_ORDER_TT0B`

`LV_LCD_BIT_PAGE_ADDRESS_ORDER_BT0T`

`LV_LCD_GAMMA_2_2`

`LV_LCD_GAMMA_1_8`

`LV_LCD_GAMMA_2_5`

`LV_LCD_GAMMA_1_0`

`LV_LCD_PIXEL_FORMAT_RGB565`

`LV_LCD_PIXEL_FORMAT_RGB666`

`LV_LCD_FLAG_NONE`

`LV_LCD_FLAG_MIRROR_X`

`LV_LCD_FLAG_MIRROR_Y`

`LV_LCD_FLAG_BGR`

`LV_LCD_FLAG_RGB666`

`LV_LCD_CMD_DELAY_MS`

`LV_LCD_CMD_EOF`

## Typedef

`typedef uint32_t lv_lcd_flag_t`

Configuration flags for `lv_lcd_xxx_create()`

`typedef void (*lv_lcd_send_cmd_cb_t)(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, const uint8_t *param, size_t param_size)`

Prototype of a platform-dependent callback to transfer commands and data to the LCD controller.

**Param disp**

display object

**Param cmd**

command buffer (can handle 16 bit commands as well)

**Param cmd\_size**

number of bytes of the command

**Param param**

parameter buffer

**Param param\_size**

number of bytes of the parameters

`typedef void (*lv_lcd_send_color_cb_t)(lv_display_t *disp, const uint8_t *cmd, size_t cmd_size, uint8_t *param, size_t param_size)`

Prototype of a platform-dependent callback to transfer pixel data to the LCD controller.

**Param disp**

display object

**Param cmd**

command buffer (can handle 16 bit commands as well)

**Param cmd\_size**

number of bytes of the command

**Param param**

parameter buffer

**Param param\_size**

number of bytes of the parameters

## Functions

`lv_display_t *lv_lcd_generic_mipi_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags, lv_lcd_send_cmd_cb_t send_cmd_cb, lv_lcd_send_color_cb_t send_color_cb)`

Create a MIPI DCS compatible LCD display

**Parametri**

- **hor\_res** -- horizontal resolution
- **ver\_res** -- vertical resolution
- **flags** -- default configuration settings (mirror, RGB ordering, etc.)
- **send\_cmd** -- platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
- **send\_color** -- platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer). `lv_display_flush_ready` must be called after the transfer has finished.

**Ritorna**

pointer to the created display

`void lv_lcd_generic_mipi_set_gap(lv_display_t *disp, uint16_t x, uint16_t y)`

Set gap, i.e., the offset of the (0,0) pixel in the VRAM

**Parametri**

- **disp** -- display object
- **x** -- x offset
- **y** -- y offset

```
void lv_lcd_generic_mipi_set_invert(lv_display_t *disp, bool invert)
    Set color inversion
    Parametri
        • disp -- display object
        • invert -- false: normal, true: invert

void lv_lcd_generic_mipi_set_address_mode(lv_display_t *disp, bool mirror_x, bool mirror_y,
                                         bool swap_xy, bool bgr)
    Set address mode
    Parametri
        • disp -- display object
        • mirror_x -- horizontal mirror (false: normal, true: mirrored)
        • mirror_y -- vertical mirror (false: normal, true: mirrored)
        • swap_xy -- swap axes (false: normal, true: swap)
        • bgr -- RGB/BGR order (false: RGB, true: BGR)

void lv_lcd_generic_mipi_set_gamma_curve(lv_display_t *disp, uint8_t gamma)
    Set gamma curve
    Parametri
        • disp -- display object
        • gamma -- gamma curve

void lv_lcd_generic_mipi_send_cmd_list(lv_display_t *disp, const uint8_t *cmd_list)
    Send list of commands.
    Parametri
        • disp -- display object
        • cmd_list -- controller and panel-specific commands
```

```
struct lv_lcd_generic_mipi_driver_t
#include <lv_lcd_generic_mipi.h> Generic MIPI compatible LCD driver
```

### Public Members

lv\_display\_t \***disp**

*lv\_lcd\_send\_cmd\_cb\_t* **send\_cmd**

*lv\_lcd\_send\_color\_cb\_t* **send\_color**

uint16\_t **x\_gap**

uint16\_t **y\_gap**

uint8\_t **madctl\_reg**

uint8\_t **colmod\_reg**

bool **mirror\_x**

bool **mirror\_y**

bool **swap\_xy**

**nxp\_elcdif****lv\_nxp\_elcdif.h**

Driver for NXP's ELCD

**Functions**

```
lv_display_t *lv_nxp_display_elcdif_create_direct(LCDIF_Type *base, const
                                                 elcdif_rgb_mode_config_t *config, void
                                                 *frame_buffer1, void *frame_buffer2, size_t
                                                 buf_size)
```

Attach LVGL to ELCDIF using DIRECT rendering mode. ELCDIF should be already initialized.

**Parametri**

- **base** -- The NXP eLCD controller base address
- **config** -- NXP eLCD config object
- **frame\_buffer1** -- pointer the first frame buffers
- **frame\_buffer2** -- pointer the second frame buffers
- **buf\_size** -- size of a buffer in bytes (must be at least as large as the screen)

**Ritorna**

a display object initialized and registered on the LVGL runtime

```
lv_display_t *lv_nxp_display_elcdif_create_partial(LCDIF_Type *base, const
                                                 elcdif_rgb_mode_config_t *config, void
                                                 *frame_buffer1, void *frame_buffer2, size_t
                                                 buf_size)
```

Attach LVGL to ELCDIF using PARTIAL rendering mode. ELCDIF should be already initialized.

**Parametri**

- **base** -- The NXP eLCD controller base address
- **config** -- NXP eLCD config object
- **frame\_buffer1** -- pointer the first frame buffers
- **frame\_buffer2** -- pointer the second frame buffers
- **buf\_size** -- size of a buffer in bytes

**Ritorna**

a display object initialized and registered on the LVGL runtime

**void lv\_nxp\_display\_elcdif\_event\_handler(const lv\_display\_t \*disp)**

Call this function on the LCD Interrupt Service Routine It tells to LVGL what to do when a framebuffer is transmitted to the LCD panel

**Nota**

: the parameter disp is typically the return value after `lv_nxp_display_elcdif_create_direct` has been successfully executed

**Parametri**

**disp** -- The display instance that contains the eLCD related data

**renesas\_glcde****lv\_renesas\_glcde.h****Functions**

```
lv_display_t *lv_renesas_glcde_direct_create(void)
```

Create a display using Renesas' GLCDC peripheral in DIRECT render mode

**Ritorna**

pointer to the created display

`lv_display_t *lv_renesas_glcdfc_partial_create(void *buf1, void *buf2, size_t buf_size)`

Create a display using Renesas' GLCDC peripheral in PARTIAL render mode

#### Parametri

- **buf1** -- first buffer
- **buf2** -- second buffer (can be NULL)
- **buf\_size** -- buffer size in byte

#### Ritorna

pointer to the created display

## st7735

### lv\_st7735.h

This driver is just a wrapper around the generic MIPI compatible LCD controller driver

#### Typedef

`typedef lv_lcd_send_cmd_cb_t lv_st7735_send_cmd_cb_t`

`typedef lv_lcd_send_color_cb_t lv_st7735_send_color_cb_t`

#### Functions

`lv_display_t *lv_st7735_create(uint32_t hor_res, uint32_t ver_res, lv_flag_t flags,  
 lv_st7735_send_cmd_cb_t send_cmd_cb, lv_st7735_send_color_cb_t  
 send_color_cb)`

Create an LCD display with ST7735 driver

#### Parametri

- **hor\_res** -- horizontal resolution
- **ver\_res** -- vertical resolution
- **flags** -- default configuration settings (mirror, RGB ordering, etc.)
- **send\_cmd** -- platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
- **send\_color** -- platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' callback)

#### Ritorna

pointer to the created display

`void lv_st7735_set_gap(lv_display_t *disp, uint16_t x, uint16_t y)`

Set gap, i.e., the offset of the (0,0) pixel in the VRAM

#### Parametri

- **disp** -- display object
- **x** -- x offset
- **y** -- y offset

`void lv_st7735_set_invert(lv_display_t *disp, bool invert)`

Set color inversion

#### Parametri

- **disp** -- display object
- **invert** -- false: normal, true: invert

`void lv_st7735_set_gamma_curve(lv_display_t *disp, uint8_t gamma)`

Set gamma curve

#### Parametri

- **disp** -- display object
- **gamma** -- gamma curve

---

```
void lv_st7735_send_cmd_list(lv_display_t *disp, const uint8_t *cmd_list)
```

Send list of commands.

**Parametri**

- **disp** -- display object
- **cmd\_list** -- controller and panel-specific commands

**st7789**

**lv\_st7789.h**

This driver is just a wrapper around the generic MIPI compatible LCD controller driver

**Typedef**

```
typedef lv_lcd_send_cmd_cb_t lv_st7789_send_cmd_cb_t
```

```
typedef lv_lcd_send_color_cb_t lv_st7789_send_color_cb_t
```

**Functions**

```
lv_display_t *lv_st7789_create(uint32_t hor_res, uint32_t ver_res, lv_lcd_flag_t flags,
                               lv_st7789_send_cmd_cb_t send_cmd_cb, lv_st7789_send_color_cb_t
                               send_color_cb)
```

Create an LCD display with ST7789 driver

**Parametri**

- **hor\_res** -- horizontal resolution
- **ver\_res** -- vertical resolution
- **flags** -- default configuration settings (mirror, RGB ordering, etc.)
- **send\_cmd** -- platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
- **send\_color** -- platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' callback)

**Ritorna**

pointer to the created display

```
void lv_st7789_set_gap(lv_display_t *disp, uint16_t x, uint16_t y)
```

Set gap, i.e., the offset of the (0,0) pixel in the VRAM

**Parametri**

- **disp** -- display object
- **x** -- x offset
- **y** -- y offset

```
void lv_st7789_set_invert(lv_display_t *disp, bool invert)
```

Set color inversion

**Parametri**

- **disp** -- display object
- **invert** -- false: normal, true: invert

```
void lv_st7789_set_gamma_curve(lv_display_t *disp, uint8_t gamma)
```

Set gamma curve

**Parametri**

- **disp** -- display object
- **gamma** -- gamma curve

```
void lv_st7789_send_cmd_list(lv_display_t *disp, const uint8_t *cmd_list)
```

Send list of commands.

**Parametri**

- **disp** -- display object

- **cmd\_list** -- controller and panel-specific commands

**st7796****lv\_st7796.h**

This driver is just a wrapper around the generic MIPI compatible LCD controller driver

**Typedef**

typedef *lv\_lcd\_send\_cmd\_cb\_t* **lv\_st7796\_send\_cmd\_cb\_t**

typedef *lv\_lcd\_send\_color\_cb\_t* **lv\_st7796\_send\_color\_cb\_t**

**Functions**

**lv\_display\_t \*lv\_st7796\_create**(uint32\_t hor\_res, uint32\_t ver\_res, *lv\_lcd\_flag\_t* flags,  
*lv\_st7796\_send\_cmd\_cb\_t* send\_cmd\_cb, *lv\_st7796\_send\_color\_cb\_t*  
send\_color\_cb)

Create an LCD display with ST7796 driver

**Parametri**

- **hor\_res** -- horizontal resolution
- **ver\_res** -- vertical resolution
- **flags** -- default configuration settings (mirror, RGB ordering, etc.)
- **send\_cmd** -- platform-dependent function to send a command to the LCD controller (usually uses polling transfer)
- **send\_color** -- platform-dependent function to send pixel data to the LCD controller (usually uses DMA transfer: must implement a 'ready' callback)

**Ritorna**

pointer to the created display

**void lv\_st7796\_set\_gap(lv\_display\_t \*disp, uint16\_t x, uint16\_t y)**

Set gap, i.e., the offset of the (0,0) pixel in the VRAM

**Parametri**

- **disp** -- display object
- **x** -- x offset
- **y** -- y offset

**void lv\_st7796\_set\_invert(lv\_display\_t \*disp, bool invert)**

Set color inversion

**Parametri**

- **disp** -- display object
- **invert** -- false: normal, true: invert

**void lv\_st7796\_set\_gamma\_curve(lv\_display\_t \*disp, uint8\_t gamma)**

Set gamma curve

**Parametri**

- **disp** -- display object
- **gamma** -- gamma curve

**void lv\_st7796\_send\_cmd\_list(lv\_display\_t \*disp, const uint8\_t \*cmd\_list)**

Send list of commands.

**Parametri**

- **disp** -- display object
- **cmd\_list** -- controller and panel-specific commands

**st\_ltcd****lv\_st\_ltcd.h****Functions**

`lv_display_t *lv_st_ltcd_create_direct(void *fb_adr_1, void *fb_adr_2, uint32_t layer_idx)`

Create a direct render mode display bound to a LTDC layer.

**Parametri**

- **fb\_adr\_1** -- The LTDC layer's framebuffer memory address.
- **fb\_adr\_2** -- An additional framebuffer-sized buffer to use for double buffering, or NULL.
- **layer\_idx** -- The LTDC layer number to bind the display to. Typically 0 or 1.

**Ritorna**

The display.

`lv_display_t *lv_st_ltcd_create_partial(void *render_buf_1, void *render_buf_2, uint32_t buf_size, uint32_t layer_idx)`

Create a partial render mode display bound to a LTDC layer. The layer's framebuffer is flushed to internally. Enable `LV_ST_LTDC_USE_DMA2D_FLUSH` for parallel flushing.

**Parametri**

- **render\_buf\_1** -- A render buffer.
- **render\_buf\_2** -- An additional render buffer for double-buffering, or NULL.
- **buf\_size** -- The size of the buffer(s) in bytes.
- **layer\_idx** -- The LTDC layer number to bind the display to. Typically 0 or 1.

**Ritorna**

The display.

**tft\_espi****lv\_tft\_espi.h****Functions**

`lv_display_t *lv_tft_espi_create(uint32_t hor_res, uint32_t ver_res, void *buf, uint32_t buf_size_bytes)`

**14.13.3 draw****eve****lv\_draw\_eve\_display.h****Functions**

`lv_display_t *lv_draw_eve_display_create(const lv_draw_eve_parameters_t *params, lv_draw_eve_operation_cb_t op_cb, void *user_data)`

Create a display for the EVE draw unit.

**Parametri**

- **params** -- Pointer to a struct of display parameters. Can be a temporary variable
- **op\_cb** -- A callback that will be called to perform pin and SPI IO operations with the EVE chip
- **user\_data** -- use `lv_draw_eve_display_get_user_data` to get this pointer inside the `op_cb`

**Ritorna**

the EVE display

`void *lv_draw_eve_display_get_user_data(lv_display_t *disp)`

Get the `user_data` parameter that was passed to `lv_draw_eve_display_create`. Useful in the operation callback.

**Parametri**

- **disp** -- pointer to the lv\_draw\_eve display

**Ritorna**  
the user\_data pointer

**lv\_indev\_t \*lv\_draw\_eve\_touch\_create(lv\_display\_t \*disp)**  
Create a touchscreen indev for the EVE display.

**Parametri**  
**disp** -- pointer to the lv\_draw\_eve display

**Ritorna**  
the EVE touchscreen indev

**uint8\_t lv\_draw\_eve\_memread8(lv\_display\_t \*disp, uint32\_t address)**  
Call EVE\_memRead8 for custom low-level control of the display.

**Parametri**

- **disp** -- the display returned by lv\_draw\_eve\_display\_create
- **address** -- the EVE address to read from

**Ritorna**  
the read value

**uint16\_t lv\_draw\_eve\_memread16(lv\_display\_t \*disp, uint32\_t address)**  
Call EVE\_memRead16 for custom low-level control of the display.

**Parametri**

- **disp** -- the display returned by lv\_draw\_eve\_display\_create
- **address** -- the EVE address to read from

**Ritorna**  
the read value

**uint32\_t lv\_draw\_eve\_memread32(lv\_display\_t \*disp, uint32\_t address)**  
Call EVE\_memRead32 for custom low-level control of the display.

**Parametri**

- **disp** -- the display returned by lv\_draw\_eve\_display\_create
- **address** -- the EVE address to read from

**Ritorna**  
the read value

**void lv\_draw\_eve\_memwrite8(lv\_display\_t \*disp, uint32\_t address, uint8\_t data)**  
Call EVE\_memWrite8 for custom low-level control of the display.

**Parametri**

- **disp** -- the display returned by lv\_draw\_eve\_display\_create
- **address** -- the EVE address to write to
- **data** -- the value to write

**void lv\_draw\_eve\_memwrite16(lv\_display\_t \*disp, uint32\_t address, uint16\_t data)**  
Call EVE\_memWrite16 for custom low-level control of the display.

**Parametri**

- **disp** -- the display returned by lv\_draw\_eve\_display\_create
- **address** -- the EVE address to write to
- **data** -- the value to write

**void lv\_draw\_eve\_memwrite32(lv\_display\_t \*disp, uint32\_t address, uint32\_t data)**  
Call EVE\_memWrite32 for custom low-level control of the display.

**Parametri**

- **disp** -- the display returned by lv\_draw\_eve\_display\_create
- **address** -- the EVE address to write to
- **data** -- the value to write

## lv\_draw\_eve\_displayDefines.h

### Defines

**LV\_EVE\_EVE\_RAM\_G**

**LV\_EVE\_EVE\_ROM\_CHIPID**

**LV\_EVE\_EVE\_ROM\_FONT**

**LV\_EVE\_EVE\_ROM\_FONTROOT**

**LV\_EVE\_EVE\_RAM\_DL**

**LV\_EVE\_EVE\_RAM\_REG**

**LV\_EVE\_EVE\_RAM\_CMD**

**LV\_EVE\_EVE\_RAM\_G\_SIZE**

**LV\_EVE\_EVE\_CMDFIFO\_SIZE**

**LV\_EVE\_EVE\_RAM\_DL\_SIZE**

**LV\_EVE\_DL\_DISPLAY**

**LV\_EVE\_DL\_BITMAP\_SOURCE**

**LV\_EVE\_DL\_CLEAR\_COLOR\_RGB**

**LV\_EVE\_DL\_TAG**

**LV\_EVE\_DL\_COLOR\_RGB**

**LV\_EVE\_DL\_BITMAP\_HANDLE**

**LV\_EVE\_DL\_CELL**

**LV\_EVE\_DL\_BITMAP\_LAYOUT**

**LV\_EVE\_DL\_BITMAP\_SIZE**

**LV\_EVE\_DL\_ALPHA\_FUNC**

**LV\_EVE\_DL\_STENCIL\_FUNC**

**LV\_EVE\_DL\_BLEND\_FUNC**

**LV\_EVE\_DL\_STENCIL\_OP**

**LV\_EVE\_DL\_POINT\_SIZE**

**LV\_EVE\_DL\_LINE\_WIDTH**

**LV\_EVE\_DL\_CLEAR\_COLOR\_A**

**LV\_EVE\_DL\_COLOR\_A**

**LV\_EVE\_DL\_CLEAR\_STENCIL**

**LV\_EVE\_DL\_CLEAR\_TAG**

**LV\_EVE\_DL\_STENCIL\_MASK**

**LV\_EVE\_DL\_TAG\_MASK**

**LV\_EVE\_DL\_BITMAP\_TRANSFORM\_A**

**LV\_EVE\_DL\_BITMAP\_TRANSFORM\_B**

**LV\_EVE\_DL\_BITMAP\_TRANSFORM\_C**

**LV\_EVE\_DL\_BITMAP\_TRANSFORM\_D**

**LV\_EVE\_DL\_BITMAP\_TRANSFORM\_E**

**LV\_EVE\_DL\_BITMAP\_TRANSFORM\_F**

**LV\_EVE\_DL\_SCISSOR\_XY**

**LV\_EVE\_DL\_SCISSOR\_SIZE**

**LV\_EVE\_DL\_CALL**

**LV\_EVE\_DL\_JUMP**

**LV\_EVE\_DL\_BEGIN**

**LV\_EVE\_DL\_COLOR\_MASK**

**LV\_EVE\_DL\_END**

**LV\_EVE\_DL\_SAVE\_CONTEXT**

**LV\_EVE\_DL\_RESTORE\_CONTEXT**

`LV_EVE_DL_RETURN`

`LV_EVE_DL_MACRO`

`LV_EVE_DL_CLEAR`

`LV_EVE_DL_VERTEX_FORMAT`

`LV_EVE_DL_BITMAP_LAYOUT_H`

`LV_EVE_DL_BITMAP_SIZE_H`

`LV_EVE_DL_PALETTE_SOURCE`

`LV_EVE_DL_VERTEX_TRANSLATE_X`

`LV_EVE_DL_VERTEX_TRANSLATE_Y`

`LV_EVE_DL_NOP`

`LV_EVE_DL_VERTEX2F`

`LV_EVE_DL_VERTEX2II`

`LV_EVE_CLR_COL`

`LV_EVE_CLR_STN`

`LV_EVE_CLR_TAG`

`LV_EVE_EVE_ACTIVE`

`LV_EVE_EVE_STANDBY`

`LV_EVE_EVE_SLEEP`

`LV_EVE_EVE_CLKEXT`

`LV_EVE_EVE_CLKINT`

`LV_EVE_EVE_PWRDOWN`

`LV_EVE_EVE_CLKSEL`

`LV_EVE_EVE_RST_PULSE`

LV\_EVE\_EVE\_CORERST  
LV\_EVE\_EVE\_PINDRIVE  
LV\_EVE\_EVE\_PIN\_PD\_STATE  
LV\_EVE\_EVE\_NEVER  
LV\_EVE\_EVE\_LESS  
LV\_EVE\_EVE\_LEQUAL  
LV\_EVE\_EVE\_GREATER  
LV\_EVE\_EVE\_GEQUAL  
LV\_EVE\_EVE\_EQUAL  
LV\_EVE\_EVE\_NOTEQUAL  
LV\_EVE\_EVE\_ALWAYS  
LV\_EVE\_EVE\_ARGB1555  
LV\_EVE\_EVE\_L1  
LV\_EVE\_EVE\_L4  
LV\_EVE\_EVE\_L8  
LV\_EVE\_EVE\_RGB332  
LV\_EVE\_EVE\_ARGB2  
LV\_EVE\_EVE\_ARGB4  
LV\_EVE\_EVE\_RGB565  
LV\_EVE\_EVE\_PALETTED  
LV\_EVE\_EVE\_TEXT8X8  
LV\_EVE\_EVE\_TEXTVGA  
LV\_EVE\_EVE\_BARGRAPH

**LV\_EVE\_EVE\_NEAREST**

**LV\_EVE\_EVE\_BILINEAR**

**LV\_EVE\_EVE\_BORDER**

**LV\_EVE\_EVE\_REPEAT**

**LV\_EVE\_EVE\_KEEP**

**LV\_EVE\_EVE\_REPLACE**

**LV\_EVE\_EVE\_INCR**

**LV\_EVE\_EVE\_DECR**

**LV\_EVE\_EVE\_INVERT**

**LV\_EVE\_EVE\_DLSWAP\_DONE**

**LV\_EVE\_EVE\_DLSWAP\_LINE**

**LV\_EVE\_EVE\_DLSWAP\_FRAME**

**LV\_EVE\_EVE\_INT\_SWAP**

**LV\_EVE\_EVE\_INT\_TOUCH**

**LV\_EVE\_EVE\_INT\_TAG**

**LV\_EVE\_EVE\_INT\_SOUND**

**LV\_EVE\_EVE\_INT\_PLAYBACK**

**LV\_EVE\_EVE\_INT\_CMDEMPTY**

**LV\_EVE\_EVE\_INT\_CMDFLAG**

**LV\_EVE\_EVE\_INT\_CONVCOMPLETE**

**LV\_EVE\_EVE\_TMODE\_OFF**

**LV\_EVE\_EVE\_TMODE\_ONESHOT**

**LV\_EVE\_EVE\_TMODE\_FRAME**

**LV\_EVE\_EVE\_TMODE\_CONTINUOUS**

**LV\_EVE\_EVE\_ZERO**

**LV\_EVE\_EVE\_ONE**

**LV\_EVE\_EVE\_SRC\_ALPHA**

**LV\_EVE\_EVE\_DST\_ALPHA**

**LV\_EVE\_EVE\_ONE\_MINUS\_SRC\_ALPHA**

**LV\_EVE\_EVE\_ONE\_MINUS\_DST\_ALPHA**

**LV\_EVE\_EVE\_BITMAPS**

**LV\_EVE\_EVE\_POINTS**

**LV\_EVE\_EVE\_LINES**

**LV\_EVE\_EVE\_LINE\_STRIP**

**LV\_EVE\_EVE\_EDGE\_STRIP\_R**

**LV\_EVE\_EVE\_EDGE\_STRIP\_L**

**LV\_EVE\_EVE\_EDGE\_STRIP\_A**

**LV\_EVE\_EVE\_EDGE\_STRIP\_B**

**LV\_EVE\_EVE\_RECTS**

**LV\_EVE\_EVE\_INT\_G8**

**LV\_EVE\_EVE\_INT\_L8C**

**LV\_EVE\_EVE\_INT\_VGA**

**LV\_EVE\_EVE\_PALETTED565**

**LV\_EVE\_EVE\_PALETTED4444**

**LV\_EVE\_EVE\_PALETTED8**

**LV\_EVE\_EVE\_L2**

**LV\_EVE\_EVE\_OPT\_MONO**

**LV\_EVE\_EVE\_OPT\_NODL**

**LV\_EVE\_EVE\_OPT\_FLAT**

**LV\_EVE\_EVE\_OPT\_CENTERX**

**LV\_EVE\_EVE\_OPT\_CENTERY**

**LV\_EVE\_EVE\_OPT\_CENTER**

**LV\_EVE\_EVE\_OPT\_NOBACK**

**LV\_EVE\_EVE\_OPT\_NOTICKS**

**LV\_EVE\_EVE\_OPT\_NOHM**

**LV\_EVE\_EVE\_OPT\_NOPOINTER**

**LV\_EVE\_EVE\_OPT\_NOSECS**

**LV\_EVE\_EVE\_OPT\_NOHANDS**

**LV\_EVE\_EVE\_OPT\_RIGHTX**

**LV\_EVE\_EVE\_OPT\_SIGNED**

**LV\_EVE\_EVE\_OPT\_MEDIAFIFO**

**LV\_EVE\_EVE\_OPT\_FULLSCREEN**

**LV\_EVE\_EVE\_OPT\_NOTEAR**

**LV\_EVE\_EVE\_OPT\_SOUND**

**LV\_EVE\_EVE\_ADC\_DIFFERENTIAL**

**LV\_EVE\_EVE\_ADC\_SINGLE\_ENDED**

**LV\_EVE\_EVE\_NUMCHAR\_PERFONT**

**LV\_EVE\_EVE\_FONT\_TABLE\_SIZE**

**LV\_EVE\_EVE\_FONT\_TABLE\_POINTER**

**LV\_EVE\_EVE\_LINEAR\_SAMPLES**

**LV\_EVE\_EVE\_ULAW\_SAMPLES**

**LV\_EVE\_EVE\_ADPCM\_SAMPLES**

**LV\_EVE\_EVE\_SILENCE**

**LV\_EVE\_EVE\_SQUAREWAVE**

**LV\_EVE\_EVE\_SINEWAVE**

**LV\_EVE\_EVE\_SAWTOOTH**

**LV\_EVE\_EVE\_TRIANGLE**

**LV\_EVE\_EVE\_BEEPING**

**LV\_EVE\_EVE\_ALARM**

**LV\_EVE\_EVE\_WARBLE**

**LV\_EVE\_EVE\_CAROUSEL**

**LV\_EVE\_EVE\_PIPS(n)**

**LV\_EVE\_EVE\_HARP**

**LV\_EVE\_EVE\_XYLOPHONE**

**LV\_EVE\_EVE\_TUBA**

**LV\_EVE\_EVE\_GLOCKENSPIEL**

**LV\_EVE\_EVE\_ORGAN**

**LV\_EVE\_EVE\_TRUMPET**

**LV\_EVE\_EVE\_PIANO**

**LV\_EVE\_EVE\_CHIMES**

**LV\_EVE\_EVE\_MUSICBOX**

**LV\_EVE\_EVE\_BELL**

**LV\_EVE\_EVE\_CLICK**

**LV\_EVE\_EVE\_SWITCH**

**LV\_EVE\_EVE\_COWBELL**

**LV\_EVE\_EVE\_NOTCH**

**LV\_EVE\_EVE\_HIHAT**

**LV\_EVE\_EVE\_KICKDRUM**

**LV\_EVE\_EVE\_POP**

**LV\_EVE\_EVE\_CLACK**

**LV\_EVE\_EVE\_CHACK**

**LV\_EVE\_EVE\_MUTE**

**LV\_EVE\_EVE\_UNMUTE**

**LV\_EVE\_EVE\_MIDI\_A0**

**LV\_EVE\_EVE\_MIDI\_A\_0**

**LV\_EVE\_EVE\_MIDI\_B0**

**LV\_EVE\_EVE\_MIDI\_C1**

**LV\_EVE\_EVE\_MIDI\_C\_1**

**LV\_EVE\_EVE\_MIDI\_D1**

**LV\_EVE\_EVE\_MIDI\_D\_1**

**LV\_EVE\_EVE\_MIDI\_E1**

**LV\_EVE\_EVE\_MIDI\_F1**

**LV\_EVE\_EVE\_MIDI\_F\_1**

**LV\_EVE\_EVE\_MIDI\_G1**

**LV\_EVE\_EVE\_MIDI\_G\_1**

**LV\_EVE\_EVE\_MIDI\_A1**

**LV\_EVE\_EVE\_MIDI\_A\_1**

**LV\_EVE\_EVE\_MIDI\_B1**

**LV\_EVE\_EVE\_MIDI\_C2**

**LV\_EVE\_EVE\_MIDI\_C\_2**

**LV\_EVE\_EVE\_MIDI\_D2**

**LV\_EVE\_EVE\_MIDI\_D\_2**

**LV\_EVE\_EVE\_MIDI\_E2**

**LV\_EVE\_EVE\_MIDI\_F2**

**LV\_EVE\_EVE\_MIDI\_F\_2**

**LV\_EVE\_EVE\_MIDI\_G2**

**LV\_EVE\_EVE\_MIDI\_G\_2**

**LV\_EVE\_EVE\_MIDI\_A2**

**LV\_EVE\_EVE\_MIDI\_A\_2**

**LV\_EVE\_EVE\_MIDI\_B2**

**LV\_EVE\_EVE\_MIDI\_C3**

**LV\_EVE\_EVE\_MIDI\_C\_3**

**LV\_EVE\_EVE\_MIDI\_D3**

**LV\_EVE\_EVE\_MIDI\_D\_3**

**LV\_EVE\_EVE\_MIDI\_E3**

**LV\_EVE\_EVE\_MIDI\_F3**

**LV\_EVE\_EVE\_MIDI\_F\_3**

**LV\_EVE\_EVE\_MIDI\_G3**

**LV\_EVE\_EVE\_MIDI\_G\_3**

**LV\_EVE\_EVE\_MIDI\_A3**

**LV\_EVE\_EVE\_MIDI\_A\_3**

**LV\_EVE\_EVE\_MIDI\_B3**

**LV\_EVE\_EVE\_MIDI\_C4**

**LV\_EVE\_EVE\_MIDI\_C\_4**

**LV\_EVE\_EVE\_MIDI\_D4**

**LV\_EVE\_EVE\_MIDI\_D\_4**

**LV\_EVE\_EVE\_MIDI\_E4**

**LV\_EVE\_EVE\_MIDI\_F4**

**LV\_EVE\_EVE\_MIDI\_F\_4**

**LV\_EVE\_EVE\_MIDI\_G4**

**LV\_EVE\_EVE\_MIDI\_G\_4**

**LV\_EVE\_EVE\_MIDI\_A4**

**LV\_EVE\_EVE\_MIDI\_A\_4**

**LV\_EVE\_EVE\_MIDI\_B4**

**LV\_EVE\_EVE\_MIDI\_C5**

**LV\_EVE\_EVE\_MIDI\_C\_5**

**LV\_EVE\_EVE\_MIDI\_D5**

**LV\_EVE\_EVE\_MIDI\_D\_5**

**LV\_EVE\_EVE\_MIDI\_E5**

**LV\_EVE\_EVE\_MIDI\_F5**

**LV\_EVE\_EVE\_MIDI\_F\_5**

**LV\_EVE\_EVE\_MIDI\_G5**

**LV\_EVE\_EVE\_MIDI\_G\_5**

**LV\_EVE\_EVE\_MIDI\_A5**

**LV\_EVE\_EVE\_MIDI\_A\_5**

**LV\_EVE\_EVE\_MIDI\_B5**

**LV\_EVE\_EVE\_MIDI\_C6**

**LV\_EVE\_EVE\_MIDI\_C\_6**

**LV\_EVE\_EVE\_MIDI\_D6**

**LV\_EVE\_EVE\_MIDI\_D\_6**

**LV\_EVE\_EVE\_MIDI\_E6**

**LV\_EVE\_EVE\_MIDI\_F6**

**LV\_EVE\_EVE\_MIDI\_F\_6**

**LV\_EVE\_EVE\_MIDI\_G6**

**LV\_EVE\_EVE\_MIDI\_G\_6**

**LV\_EVE\_EVE\_MIDI\_A6**

**LV\_EVE\_EVE\_MIDI\_A\_6**

**LV\_EVE\_EVE\_MIDI\_B6**

**LV\_EVE\_EVE\_MIDI\_C7**

**LV\_EVE\_EVE\_MIDI\_C\_7**

**LV\_EVE\_EVE\_MIDI\_D7**

**LV\_EVE\_EVE\_MIDI\_D\_7**

**LV\_EVE\_EVE\_MIDI\_E7**

**LV\_EVE\_EVE\_MIDI\_F7**

**LV\_EVE\_EVE\_MIDI\_F\_7**

**LV\_EVE\_EVE\_MIDI\_G7**

**LV\_EVE\_EVE\_MIDI\_G\_7**

**LV\_EVE\_EVE\_MIDI\_A7**

**LV\_EVE\_EVE\_MIDI\_A\_7**

**LV\_EVE\_EVE\_MIDI\_B7**

**LV\_EVE\_EVE\_MIDI\_C8**

**LV\_EVE\_EVE\_GPIO00**

**LV\_EVE\_EVE\_GPIO01**

**LV\_EVE\_EVE\_GPIO07**

**LV\_EVE\_EVE\_DISPLAY\_0**

**LV\_EVE\_EVE\_DISPLAY\_180**

**LV\_EVE\_CMD\_APPEND**

**LV\_EVE\_CMD\_BGCOLOR**

**LV\_EVE\_CMD\_BUTTON**

**LV\_EVE\_CMD\_CALIBRATE**

**LV\_EVE\_CMD\_CLOCK**

**LV\_EVE\_CMD\_COLDSTART**

**LV\_EVE\_CMD\_DIAL**

**LV\_EVE\_CMD\_DLSTART**

**LV\_EVE\_CMD\_FGCOLOR**

**LV\_EVE\_CMD\_GAUGE**

**LV\_EVE\_CMD\_GETMATRIX**

**LV\_EVE\_CMD\_GETPROPS**

**LV\_EVE\_CMD\_GETPTR**

**LV\_EVE\_CMD\_GRADCOLOR**

**LV\_EVE\_CMD\_GRADIENT**

**LV\_EVE\_CMD\_INFLATE**

**LV\_EVE\_CMD\_INTERRUPT**

**LV\_EVE\_CMD\_KEYS**

**LV\_EVE\_CMD\_LOADIDENTITY**

**LV\_EVE\_CMD\_LOADIMAGE**

**LV\_EVE\_CMD\_LOGO**

**LV\_EVE\_CMD\_MEDIAFIFO**

**LV\_EVE\_CMD\_MEMCPY**

**LV\_EVE\_CMD\_MEMCRC**

**LV\_EVE\_CMD\_MEMSET**

**LV\_EVE\_CMD\_MEMWRITE**

**LV\_EVE\_CMD\_MEMZERO**

**LV\_EVE\_CMD\_NUMBER**

**LV\_EVE\_CMD\_PLAYVIDEO**

**LV\_EVE\_CMD\_PROGRESS**

**LV\_EVE\_CMD\_REGREAD**

**LV\_EVE\_CMD\_ROMFONT**

**LV\_EVE\_CMD\_ROTATE**

**LV\_EVE\_CMD\_SCALE**

**LV\_EVE\_CMD\_SCREENSAVER**

**LV\_EVE\_CMD\_SCROLLBAR**

**LV\_EVE\_CMD\_SETBASE**

**LV\_EVE\_CMD\_SETBITMAP**

**LV\_EVE\_CMD\_SETFONT**

**LV\_EVE\_CMD\_SETFONT2**

**LV\_EVE\_CMD\_SETMATRIX**

**LV\_EVE\_CMD\_SETROTATE**

**LV\_EVE\_CMD\_SETSCRATCH**

**LV\_EVE\_CMD\_SKETCH**

**LV\_EVE\_CMD\_SLIDER**

**LV\_EVE\_CMD\_SNAPSHOT**

**LV\_EVE\_CMD\_SNAPSHOT2**

**LV\_EVE\_CMD\_SPINNER**

**LV\_EVE\_CMD\_STOP**

**LV\_EVE\_CMD\_SWAP**

**LV\_EVE\_CMD\_TEXT**

**LV\_EVE\_CMD\_TOGGLE**

**LV\_EVE\_CMD\_TRACK**

**LV\_EVE\_CMD\_TRANSLATE**

**LV\_EVE\_CMD\_VIDEOFRAME**

**LV\_EVE\_CMD\_VIDEOSTART**

**LV\_EVE\_REG\_ANA\_COMP**

**LV\_EVE\_REG\_BIST\_EN**

**LV\_EVE\_REG\_CLOCK**

**LV\_EVE\_REG\_CMDB\_SPACE**

**LV\_EVE\_REG\_CMDB\_WRITE**

**LV\_EVE\_REG\_CMD\_DL**

**LV\_EVE\_REG\_CMD\_READ**

**LV\_EVE\_REG\_CMD\_WRITE**

**LV\_EVE\_REG\_CPURESET**

**LV\_EVE\_REG\_CSPREAD**

**LV\_EVE\_REG\_CTOUCH\_EXTENDED**

**LV\_EVE\_REG\_CTOUCH\_TOUCH0\_XY**

**LV\_EVE\_REG\_CTOUCH\_TOUCH4\_X**

**LV\_EVE\_REG\_CTOUCH\_TOUCH4\_Y**

**LV\_EVE\_REG\_CTOUCH\_TOUCH1\_XY**

**LV\_EVE\_REG\_CTOUCH\_TOUCH2\_XY**

**LV\_EVE\_REG\_CTOUCH\_TOUCH3\_XY**

**LV\_EVE\_REG\_TOUCH\_CONFIG**

**LV\_EVE\_REG\_DATESTAMP**

**LV\_EVE\_REG\_DITHER**

**LV\_EVE\_REG\_DL\_SWAP**

**LV\_EVE\_REG\_FRAMES**

**LV\_EVE\_REG\_FREQUENCY**

**LV\_EVE\_REG\_GPIO**

`LV_EVE_REG_GPIOX`

`LV_EVE_REG_GPIOX_DIR`

`LV_EVE_REG_GPIO_DIR`

`LV_EVE_REG_HCYCLE`

`LV_EVE_REG_HOFFSET`

`LV_EVE_REG_HSIZE`

`LV_EVE_REG_HSYNC0`

`LV_EVE_REG_HSYNC1`

`LV_EVE_REG_ID`

`LV_EVE_REG_INT_EN`

`LV_EVE_REG_INT_FLAGS`

`LV_EVE_REG_INT_MASK`

`LV_EVE_REG_MACRO_0`

`LV_EVE_REG_MACRO_1`

`LV_EVE_REG_MEDIAFIFO_READ`

`LV_EVE_REG_MEDIAFIFO_WRITE`

`LV_EVE_REG_OUTBITS`

`LV_EVE_REG_PCLK`

`LV_EVE_REG_PCLK_POL`

`LV_EVE_REG_PLAY`

`LV_EVE_REG_PLAYBACK_FORMAT`

`LV_EVE_REG_PLAYBACK_FREQ`

`LV_EVE_REG_PLAYBACK_LENGTH`

**LV\_EVE\_REG\_PLAYBACK\_LOOP**

**LV\_EVE\_REG\_PLAYBACK\_PLAY**

**LV\_EVE\_REG\_PLAYBACK\_READPTR**

**LV\_EVE\_REG\_PLAYBACK\_START**

**LV\_EVE\_REG\_PWM\_DUTY**

**LV\_EVE\_REG\_PWM\_HZ**

**LV\_EVE\_REG\_RENDERMODE**

**LV\_EVE\_REG\_ROTATE**

**LV\_EVE\_REG\_SNAPFORMAT**

**LV\_EVE\_REG\_SNAPSHOT**

**LV\_EVE\_REG\_SNAPY**

**LV\_EVE\_REG\_SOUND**

**LV\_EVE\_REG\_SPI\_WIDTH**

**LV\_EVE\_REG\_SWIZZLE**

**LV\_EVE\_REG\_TAG**

**LV\_EVE\_REG\_TAG\_X**

**LV\_EVE\_REG\_TAG\_Y**

**LV\_EVE\_REG\_TAP\_CRC**

**LV\_EVE\_REG\_TAP\_MASK**

**LV\_EVE\_REG\_TOUCH\_ADC\_MODE**

**LV\_EVE\_REG\_TOUCH\_CHARGE**

**LV\_EVE\_REG\_TOUCH\_DIRECT\_XY**

**LV\_EVE\_REG\_TOUCH\_DIRECT\_Z1Z2**

**LV\_EVE\_REG\_TOUCH\_MODE**

**LV\_EVE\_REG\_TOUCH\_OVERSAMPLE**

**LV\_EVE\_REG\_TOUCH\_RAW\_XY**

**LV\_EVE\_REG\_TOUCH\_RZ**

**LV\_EVE\_REG\_TOUCH\_RZTHRESH**

**LV\_EVE\_REG\_TOUCH\_SCREEN\_XY**

**LV\_EVE\_REG\_TOUCH\_SETTLE**

**LV\_EVE\_REG\_TOUCH\_TAG**

**LV\_EVE\_REG\_TOUCH\_TAG1**

**LV\_EVE\_REG\_TOUCH\_TAG1\_XY**

**LV\_EVE\_REG\_TOUCH\_TAG2**

**LV\_EVE\_REG\_TOUCH\_TAG2\_XY**

**LV\_EVE\_REG\_TOUCH\_TAG3**

**LV\_EVE\_REG\_TOUCH\_TAG3\_XY**

**LV\_EVE\_REG\_TOUCH\_TAG4**

**LV\_EVE\_REG\_TOUCH\_TAG4\_XY**

**LV\_EVE\_REG\_TOUCH\_TAG\_XY**

**LV\_EVE\_REG\_TOUCH\_TRANSFORM\_A**

**LV\_EVE\_REG\_TOUCH\_TRANSFORM\_B**

**LV\_EVE\_REG\_TOUCH\_TRANSFORM\_C**

**LV\_EVE\_REG\_TOUCH\_TRANSFORM\_D**

**LV\_EVE\_REG\_TOUCH\_TRANSFORM\_E**

**LV\_EVE\_REG\_TOUCH\_TRANSFORM\_F**

**LV\_EVE\_REG\_TRACKER**  
**LV\_EVE\_REG\_TRACKER\_1**  
**LV\_EVE\_REG\_TRACKER\_2**  
**LV\_EVE\_REG\_TRACKER\_3**  
**LV\_EVE\_REG\_TRACKER\_4**  
**LV\_EVE\_REG\_TRIM**  
**LV\_EVE\_REG\_VCYCLE**  
**LV\_EVE\_REG\_VOFFSET**  
**LV\_EVE\_REG\_VOL\_PB**  
**LV\_EVE\_REG\_VOL\_SOUND**  
**LV\_EVE\_REG\_VSIZE**  
**LV\_EVE\_REG\_VSYNC0**  
**LV\_EVE\_REG\_VSYNC1**  
**LV\_EVE\_EVE\_GLFORMAT**  
**LV\_EVE\_DL\_BITMAP\_EXT\_FORMAT**  
**LV\_EVE\_DL\_BITMAP\_SWIZZLE**  
**LV\_EVE\_EVE\_ASTC\_4X4**  
**LV\_EVE\_EVE\_ASTC\_5X4**  
**LV\_EVE\_EVE\_ASTC\_5X5**  
**LV\_EVE\_EVE\_ASTC\_6X5**  
**LV\_EVE\_EVE\_ASTC\_6X6**  
**LV\_EVE\_EVE\_ASTC\_8X5**  
**LV\_EVE\_EVE\_ASTC\_8X6**

**LV\_EVE\_EVE\_ASTC\_8X8**

**LV\_EVE\_EVE\_ASTC\_10X5**

**LV\_EVE\_EVE\_ASTC\_10X6**

**LV\_EVE\_EVE\_ASTC\_10X8**

**LV\_EVE\_EVE\_ASTC\_10X10**

**LV\_EVE\_EVE\_ASTC\_12X10**

**LV\_EVE\_EVE\_ASTC\_12X12**

**LV\_EVE\_EVE\_RAM\_ERR\_REPORT**

**LV\_EVE\_EVE\_RAM\_FLASH**

**LV\_EVE\_EVE\_RAM\_FLASH\_POSTBLOB**

**LV\_EVE\_EVE\_OPT\_FLASH**

**LV\_EVE\_EVE\_OPT\_OVERLAY**

**LV\_EVE\_EVE\_OPT\_FORMAT**

**LV\_EVE\_EVE\_OPT\_FILL**

**LV\_EVE\_CMD\_BITMAP\_TRANSFORM**

**LV\_EVE\_CMD\_SYNC**

**LV\_EVE\_CMD\_FLASHERASE**

**LV\_EVE\_CMD\_FLASHWRITE**

**LV\_EVE\_CMD\_FLASHREAD**

**LV\_EVE\_CMD\_FLASHUPDATE**

**LV\_EVE\_CMD\_FLASHDETACH**

**LV\_EVE\_CMD\_FLASHATTACH**

**LV\_EVE\_CMD\_FLASHFAST**

**LV\_EVE\_CMD\_FLASHSPIDESEL**

**LV\_EVE\_CMD\_FLASHSPITX**

**LV\_EVE\_CMD\_FLASHSPIRX**

**LV\_EVE\_CMD\_FLASHSOURCE**

**LV\_EVE\_CMD\_CLEARCACHE**

**LV\_EVE\_CMD\_INFLATE2**

**LV\_EVE\_CMD\_ROTATEAROUND**

**LV\_EVE\_CMD\_RESETFONTS**

**LV\_EVE\_CMD\_ANIMSTART**

**LV\_EVE\_CMD\_ANIMSTOP**

**LV\_EVE\_CMD\_ANIMXY**

**LV\_EVE\_CMD\_ANIMDRAW**

**LV\_EVE\_CMD\_GRADIENTA**

**LV\_EVE\_CMD\_FILLWIDTH**

**LV\_EVE\_CMD\_APPENDF**

**LV\_EVE\_CMD\_ANIMFRAME**

**LV\_EVE\_CMD\_VIDEOSTARTF**

**LV\_EVE\_REG\_ADAPTIVE\_FRAMERATE**

**LV\_EVE\_REG\_PLAYBACK\_PAUSE**

**LV\_EVE\_REG\_FLASH\_STATUS**

**LV\_EVE\_REG\_FLASH\_SIZE**

**LV\_EVE\_REG\_PLAY\_CONTROL**

**LV\_EVE\_REG\_COPRO\_PATCH\_PTR**

**LV\_EVE\_CMD\_ANIMFRAMERAM**

**LV\_EVE\_CMD\_ANIMSTARTRAM**

**LV\_EVE\_CMD\_APILEVEL**

**LV\_EVE\_CMD\_CALIBRATESUB**

**LV\_EVE\_CMD\_CALLLIST**

**LV\_EVE\_CMD\_ENDLIST**

**LV\_EVE\_CMD\_FLASHPROGRAM**

**LV\_EVE\_CMD\_FONTCACHE**

**LV\_EVE\_CMD\_FONTCACHEQUERY**

**LV\_EVE\_CMD\_GETIMAGE**

**LV\_EVE\_CMD\_HSF**

**LV\_EVE\_CMD\_LINETIME**

**LV\_EVE\_CMD\_NEWLIST**

**LV\_EVE\_CMD\_PCLKFREQ**

**LV\_EVE\_CMD\_RETURN**

**LV\_EVE\_CMD\_RUNANIM**

**LV\_EVE\_CMD\_TESTCARD**

**LV\_EVE\_CMD\_WAIT**

**LV\_EVE\_REG\_UNDERRUN**

**LV\_EVE\_REG\_AH\_HCYCLE\_MAX**

**LV\_EVE\_REG\_PCLK\_FREQ**

**LV\_EVE\_REG\_PCLK\_2X**

**LV\_EVE\_REG\_ANIM\_ACTIVE**

## Functions

static inline uint32\_t **LV\_EVE\_ALPHA\_FUNC**(uint8\_t func, uint8\_t ref)

Set the alpha test function.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_HANDLE**(uint8\_t handle)

Set the bitmap handle.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_LAYOUT**(uint8\_t format, uint16\_t linestride, uint16\_t height)

Set the source bitmap memory format and layout for the current handle.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_SIZE**(uint8\_t filter, uint8\_t wrapx, uint8\_t wraph, uint16\_t width, uint16\_t height)

Set the source bitmap memory format and layout for the current handle.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_LAYOUT\_H**(uint16\_t linestride, uint16\_t height)

Set the 2 most significant bits of the source bitmap memory format and layout for the current handle.

**Nota**

this is different to FTDIs implementation as this takes the original values as parameters and not only the upper bits

**Parametri**

- **linestride** -- 12-bit value specified to BITMAP\_LAYOUT
- **height** -- 11-bit value specified to BITMAP\_LAYOUT

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_SIZE\_H**(uint16\_t width, uint16\_t height)

Set the 2 most significant bits of bitmaps dimension for the current handle.

**Nota**

this is different to FTDIs implementation as this takes the original values as parameters and not only the upper bits

**Parametri**

- **linestride** -- 11-bit value of bitmap width, the 2 most significant bits are used
- **height** -- 11-bit value of bitmap width, the 2 most significant bits are used

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_SOURCE**(uint32\_t addr)

Set the source address of bitmap data in RAM\_G or flash memory.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_TRANSFORM\_C**(uint32\_t val)

Set the C coefficient of the bitmap transform matrix.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_BITMAP\_TRANSFORM\_F (uint32\_t val)**

Set the F coefficient of the bitmap transform matrix.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_BLEND\_FUNC (uint8\_t src, uint8\_t dst)**

Execute a sequence of commands at another location in the display list.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_CALL (uint16\_t dest)**

Execute a sequence of commands at another location in the display list.

**Nota**

valid range for dest is from zero to 2047

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_JUMP (uint16\_t dest)**

Execute commands at another location in the display list.

**Nota**

valid range for dest is from zero to 2047

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_CELL (uint8\_t cell)**

Set the bitmap cell number for the VERTEX2F command.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_CLEAR (uint8\_t color, uint8\_t stencil, uint8\_t tag)**

Clear buffers to preset values.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_CLEAR\_COLOR\_A (uint8\_t alpha)**

Set clear value for the alpha channel.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_CLEAR\_COLOR\_RGB (uint8\_t red, uint8\_t green, uint8\_t blue)**

Set clear values for red, green and blue channels.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_CLEAR\_STENCIL (uint8\_t val)**

Set clear value for the stencil buffer.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_CLEAR\_TAG**(uint8\_t val)

Set clear value for the tag buffer.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_COLOR\_A**(uint8\_t alpha)

Set the current color alpha.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_COLOR\_MASK**(uint8\_t red, uint8\_t green, uint8\_t blue, uint8\_t alpha)

Enable or disable writing of color components.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_COLOR\_RGB**(uint8\_t red, uint8\_t green, uint8\_t blue)

Set the current color red, green and blue.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_LINE\_WIDTH**(uint16\_t width)

Set the width of lines to be drawn with primitive LINES in 1/16 pixel precision.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_MACRO**(uint8\_t macro)

Execute a single command from a macro register.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_PALETTE\_SOURCE**(uint32\_t addr)

Set the base address of the palette.

**Nota**

2-byte alignment is required if pixel format is PALETTE4444 or PALETTE565.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_POINT\_SIZE**(uint16\_t size)

Set the radius of points to be drawn with primitive POINTS in 1/16 pixel precision.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_SCISSOR\_SIZE**(uint16\_t width, uint16\_t height)

Set the size of the scissor clip rectangle.

**Nota**

valid range for width and height is from zero to 2048

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_SCISSOR\_XY**(uint16\_t xc0, uint16\_t yc0)

Set the top left corner of the scissor clip rectangle.

**Nota**

valid range for width and height is from zero to 2047

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_STENCIL\_FUNC**(uint8\_t func, uint8\_t ref, uint8\_t mask)

Set function and reference value for stencil testing.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_STENCIL\_MASK**(uint8\_t mask)

Control the writing of individual bits in the stencil planes.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_STENCIL\_OP**(uint8\_t sfail, uint8\_t spass)

Set stencil test actions.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_TAG**(uint8\_t tagval)

Attach the tag value for the following graphics objects drawn on the screen.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_TAG\_MASK**(uint8\_t mask)

Control the writing of the tag buffer.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_VERTEX2F**(int16\_t xc0, int16\_t yc0)

Set coordinates for graphics primitives.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_VERTEX2II**(uint16\_t xc0, uint16\_t yc0, uint8\_t handle, uint8\_t cell)

Set coordinates, bitmap-handle and cell-number for graphics primitives.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_VERTEX\_FORMAT**(uint8\_t frac)

Set the precision of VERTEX2F coordinates.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_VERTEX\_TRANSLATE\_X**(int32\_t xco)

Set the vertex transformations X translation component.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_VERTEX\_TRANSLATE\_Y**(int32\_t yco)

Set the vertex transformations Y translation component.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

**static inline uint32\_t LV\_EVE\_BITMAP\_EXT\_FORMAT**(uint16\_t format)

Set the extended format of the bitmap.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_SWIZZLE**(uint8\_t red, uint8\_t green, uint8\_t blue, uint8\_t alpha)

Set the source for the red, green, blue and alpha channels of a bitmap.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_TRANSFORM\_A**(uint8\_t prc, uint32\_t val)

Set the A coefficient of the bitmap transform matrix.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_TRANSFORM\_B**(uint8\_t prc, uint32\_t val)

Set the B coefficient of the bitmap transform matrix.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_TRANSFORM\_D**(uint8\_t prc, uint32\_t val)

Set the D coefficient of the bitmap transform matrix.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

static inline uint32\_t **LV\_EVE\_BITMAP\_TRANSFORM\_E**(uint8\_t prc, uint32\_t val)

Set the E coefficient of the bitmap transform matrix.

**Ritorna**

a 32 bit word for use with EVE\_cmd\_dl()

#### 14.13.4 evdev

**lv\_evdev.h**

**Typedef**

typedef void (\***lv\_evdev\_discovery\_cb\_t**)(lv\_indev\_t \*indev, *lv\_evdev\_type\_t* type, void \*user\_data)

**Param indev**

the indev created for the newly discovered evdev

**Param type**

the type of the evdev

**Param user\_data**

a custom parameter

**Enums**

enum **lv\_evdev\_type\_t**

*Values:*

enumerator **LV\_EVDEV\_TYPE\_REL**

mice

enumerator **LV\_EVDEV\_TYPE\_ABS**

touch screens, mousepads

enumerator **LV\_EVDEV\_TYPE\_KEY**

keyboards, keypads, buttons

**Functions**

`lv_indev_t *lv_evdev_create(lv_indev_type_t indev_type, const char *dev_path)`

Create evdev input device.

**Parametri**

- **type** -- LV\_INDEV\_TYPE\_POINTER or LV\_INDEV\_TYPE\_KEYPAD
- **dev\_path** -- device path, e.g., /dev/input/event0

**Ritorna**

pointer to input device or NULL if opening failed

`lv_result_t lv_evdev_discovery_start(lv_evdev_discovery_cb_t cb, void *user_data)`

Begin automatically creating evdev indevs for all new and existing evdev devices found in /dev/input/

**Parametri**

- **cb** -- function to call when a new evdev indev is discovered, or NULL
- **user\_data** -- parameter to pass to the callback

**Ritorna**

the success or failure status. It will fail if it's already running or resources could not be initialized.

`lv_result_t lv_evdev_discovery_stop(void)`

Stop automatically creating evdev indevs. Safe to call from the discovery callback.

**Ritorna**

the success or failure status. It will fail if it's already running.

`void lv_evdev_set_swap_axes(lv_indev_t *indev, bool swap_axes)`

Set whether coordinates of pointer device should be swapped. Defaults to false.

**Parametri**

- **indev** -- evdev input device
- **swap\_axes** -- whether to swap x and y axes

`void lv_evdev_set_calibration(lv_indev_t *indev, int min_x, int min_y, int max_x, int max_y)`

Configure a coordinate transformation for pointer devices. Applied after axis swap, if any. Defaults to apply no transformation.

**Parametri**

- **indev** -- evdev input device
- **min\_x** -- pointer coordinate mapped to min x of display
- **min\_y** -- pointer coordinate mapped to min y of display
- **max\_x** -- pointer coordinate mapped to max x of display
- **max\_y** -- pointer coordinate mapped to max y of display

`void lv_evdev_delete(lv_indev_t *indev)`

Remove evdev input device.

**Parametri**

**indev** -- evdev input device to close and free

**lv\_evdev\_private.h****Functions**

`void lv_evdev_deinit(void)`

**14.13.5 glfw****lv\_glfw\_window.h****Functions**

`lv_glfw_window_t *lv_glfw_window_create(int32_t hor_res, int32_t ver_res, bool use_mouse_indev)`

Create a GLFW window with no textures and initialize OpenGL

**Parametri**

- **hor\_res** -- width in pixels of the window

- **ver\_res** -- height in pixels of the window
- **use\_mouse\_indev** -- send pointer indev input to LVGL display textures

**Ritorna**

the new GLFW window handle

**void lv\_glfw\_window\_delete(lv\_glfw\_window\_t \*window)**

Delete a GLFW window. If it is the last one, the process will exit

**Parametri**

**window** -- GLFW window to delete

**void \*lv\_glfw\_window\_get\_glfw\_window(lv\_glfw\_window\_t \*window)**

Get the GLFW window handle for an lv\_glfw\_window

**Parametri**

**window** -- GLFW window to return the handle of

**Ritorna**

the GLFW window handle

**lv\_glfw\_texture\_t \*lv\_glfw\_window\_add\_texture(lv\_glfw\_window\_t \*window, unsigned int texture\_id, int32\_t w, int32\_t h)**

Add a texture to the GLFW window. It can be an LVGL display texture, or any OpenGL texture

**Parametri**

- **window** -- GLFW window
- **texture\_id** -- OpenGL texture ID
- **w** -- width in pixels of the texture
- **h** -- height in pixels of the texture

**Ritorna**

the new texture handle

**void lv\_glfw\_texture\_remove(lv\_glfw\_texture\_t \*texture)**

Remove a texture from its GLFW window and delete it

**Parametri**

**texture** -- handle of a GLFW window texture

**void lv\_glfw\_texture\_set\_x(lv\_glfw\_texture\_t \*texture, int32\_t x)**

Set the x position of a texture within its GLFW window

**Parametri**

- **texture** -- handle of a GLFW window texture
- **x** -- new x position of the texture

**void lv\_glfw\_texture\_set\_y(lv\_glfw\_texture\_t \*texture, int32\_t y)**

Set the y position of a texture within its GLFW window

**Parametri**

- **texture** -- handle of a GLFW window texture
- **y** -- new y position of the texture

**void lv\_glfw\_texture\_set\_opa(lv\_glfw\_texture\_t \*texture, lv\_opa\_t opa)**

Set the opacity of a texture in a GLFW window

**Parametri**

- **texture** -- handle of a GLFW window texture
- **opa** -- new opacity of the texture

**lv\_indev\_t \*lv\_glfw\_texture\_get\_mouse\_indev(lv\_glfw\_texture\_t \*texture)**

Get the mouse indev associated with a texture in a GLFW window, if it exists

**Nota**

there will only be an indev if the texture is based on an LVGL display texture and the window was created with **use\_mouse\_indev** as **true**

**Parametri****texture** -- handle of a GLFW window texture**Ritorna**the indev or **NULL****lv\_glfw\_window\_private.h**struct **\_lv\_glfw\_window\_t****Public Members**GLFWwindow \***window**int32\_t **hor\_res**int32\_t **ver\_res***lv\_ll\_t* **textures***lv\_point\_t* **mouse\_last\_point***lv\_indev\_state\_t* **mouse\_last\_state**uint8\_t **use\_indev**uint8\_t **closing**struct **\_lv\_glfw\_texture\_t****Public Members**lv\_glfw\_window\_t \***window**unsigned int **texture\_id***lv\_area\_t* **area**lv\_opa\_t **opa**lv\_indev\_t \***indev***lv\_point\_t* **indev\_last\_point***lv\_indev\_state\_t* **indev\_last\_state**

**lv\_opengles\_debug.h****Defines****GL\_CALL(x)****Functions**void **GLClearError**(void)void **GLLogCall**(const char \*function, const char \*file, int line)**lv\_opengles\_driver.h****Functions**void **lv\_opengles\_init**(void)

Initialize OpenGL

**Nota**it is not necessary to call this if you use `lv_glfw_window_create`void **lv\_opengles\_deinit**(void)

Deinitialize OpenGL

**Nota**it is not necessary to call this if you use `lv_glfw_window_create`void **lv\_opengles\_render\_texture**(unsigned int texture, const *lv\_area\_t* \*texture\_area, *lv\_opa\_t* opa, int32\_t disp\_w, int32\_t disp\_h, const *lv\_area\_t* \*texture\_clip\_area, bool flip)

Render a texture

**Parametri**

- **texture** -- OpenGL texture ID
- **texture\_area** -- the area in the window to render the texture in
- **opa** -- opacity to blend the texture with existing contents
- **disp\_w** -- width of the window/framebuffer being rendered to
- **disp\_h** -- height of the window/framebuffer being rendered to

void **lv\_opengles\_render\_fill**(*lv\_color\_t* color, const *lv\_area\_t* \*area, *lv\_opa\_t* opa, int32\_t disp\_w, int32\_t disp\_h)

Render a fill

**Parametri**

- **color** -- the color of the fill
- **area** -- the area in the window to render the fill
- **opa** -- opacity to blend the fill with existing contents
- **disp\_w** -- width of the window/framebuffer being rendered to
- **disp\_h** -- height of the window/framebuffer being rendered to

void **lv\_opengles\_render\_clear**(void)

Clear the window/display

void **lv\_opengles\_viewport**(int32\_t x, int32\_t y, int32\_t w, int32\_t h)

Set the OpenGL viewport

**Parametri**

- **x** -- x position of the viewport
- **y** -- y position of the viewport
- **w** -- width of the viewport
- **h** -- height of the viewport

**lv\_opengles\_texture.h****Functions****lv\_display\_t \*lv\_opengles\_texture\_create(int32\_t w, int32\_t h)**

Create a display that flushes to an OpenGL texture

**Parametri**

- **w** -- width in pixels of the texture
- **h** -- height in pixels of the texture

**Ritorna**

the new display

**unsigned int lv\_opengles\_texture\_get\_texture\_id(lv\_display\_t \*disp)**

Get the OpenGL texture ID of the display

**Parametri****disp** -- display**Ritorna**

texture ID

**lv\_display\_t \*lv\_opengles\_texture\_get\_from\_texture\_id(unsigned int texture\_id)**

Get the display of an OpenGL texture if it is associated with one

**Parametri****texture\_id** -- OpenGL texture ID**Ritorna**

display or NULL if there no display with that texture ID

**14.13.6 libinput****lv\_libinput.h****Defines****LV\_LIBINPUT\_MAX\_EVENTS****Enums****enum lv\_libinput\_capability***Values:*enumerator **LV\_LIBINPUT\_CAPABILITY\_NONE**enumerator **LV\_LIBINPUT\_CAPABILITY\_KEYBOARD**enumerator **LV\_LIBINPUT\_CAPABILITY\_POINTER**enumerator **LV\_LIBINPUT\_CAPABILITY\_TOUCH**

## Functions

`lv_libinput_capability lv_libinput_query_capability(struct libinput_device *device)`

Determine the capabilities of a specific libinput device.

**Parametri**

**device** -- the libinput device to query

**Ritorna**

the supported input capabilities

`char *lv_libinput_find_dev(lv_libinput_capability capabilities, bool force_rescan)`

Find connected input device with specific capabilities

**Parametri**

- **capabilities** -- required device capabilities
- **force\_rescan** -- erase the device cache (if any) and rescan the file system for available devices

**Ritorna**

device node path (e.g. /dev/input/event0) for the first matching device or NULL if no device was found. The pointer is safe to use until the next forceful device search.

`size_t lv_libinput_find_devs(lv_libinput_capability capabilities, char **found, size_t count, bool force_rescan)`

Find connected input devices with specific capabilities

**Parametri**

- **capabilities** -- required device capabilities
- **devices** -- pre-allocated array to store the found device node paths (e.g. /dev/input/event0). The pointers are safe to use until the next forceful device search.
- **count** -- maximum number of devices to find (the devices array should be at least this long)
- **force\_rescan** -- erase the device cache (if any) and rescan the file system for available devices

**Ritorna**

number of devices that were found

`lv_indev_t *lv_libinput_create(lv_indev_type_t indev_type, const char *dev_path)`

Create a new libinput input device

**Parametri**

- **type** -- LV\_INDEV\_TYPE\_POINTER or LV\_INDEV\_TYPE\_KEYPAD
- **dev\_path** -- device path, e.g. /dev/input/event0

**Ritorna**

pointer to input device or NULL if opening failed

`void lv_libinput_delete(lv_indev_t *indev)`

Delete a libinput input device

**Parametri**

**indev** -- pointer to input device

## lv\_libinput\_private.h

`struct _lv_libinput_event_t`

### Public Members

`lv_indev_state_t pressed`

`int key_val`

*lv\_point\_t* **point**

struct **\_lv\_libinput\_t**

#### Public Members

int **fd**

struct pollfd **fds**[1]

lv\_libinput\_event\_t **points**[32]

lv\_libinput\_event\_t **slots**[2]

*lv\_point\_t* **pointer\_position**

bool **pointer\_button\_down**

int **start**

int **end**

lv\_libinput\_event\_t **last\_event**

bool **deinit**

pthread\_mutex\_t **event\_lock**

pthread\_t **worker\_thread**

struct libinput \***libinput\_context**

struct *libinput\_device* \***libinput\_device**

**lv\_xkb.h**

**lv\_xkb\_private.h**

**14.13.7 nuttx**

**lv\_nuttx\_cache.h**

#### Functions

void **lv\_nuttx\_cache\_init**(void)

void **lv\_nuttx\_cache\_deinit**(void)

**lv\_nuttx\_entry.h****Typedef**

```
typedef struct _lv_nuttx_ctx_t lv_nuttx_ctx_t
```

**Functions**

```
void lv_nuttx_dsc_init(lv_nuttx_dsc_t *dsc)
```

Initialize the `lv_nuttx_dsc_t` structure with default values for the NuttX port of LVGL.

**Parametri**

- dsc** -- Pointer to the `lv_nuttx_dsc_t` structure to be initialized.

```
void lv_nuttx_init(const lv_nuttx_dsc_t *dsc, lv_nuttx_result_t *result)
```

Initialize the LVGL display driver for NuttX using the provided configuration information.

**Parametri**

- dsc** -- Pointer to the `lv_nuttx_dsc_t` structure containing the configuration information for the display driver.
- result** -- Pointer to the `lv_nuttx_result_t` structure containing display and input device handler.

```
void lv_nuttx_deinit(lv_nuttx_result_t *result)
```

Deinitialize the LVGL display driver for NuttX.

**Parametri**

- result** -- Pointer to the `lv_nuttx_result_t` structure containing display and input device handler.

```
void lv_nuttx_init_custom(const lv_nuttx_dsc_t *dsc, lv_nuttx_result_t *result)
```

Initialize the LVGL display driver for NuttX using the provided custom configuration information.

**Parametri**

- dsc** -- Pointer to the `lv_nuttx_dsc_t` structure containing the custom configuration for the display driver.
- result** -- Pointer to the `lv_nuttx_result_t` structure containing display and input device handler.

```
void lv_nuttx_deinit_custom(lv_nuttx_result_t *result)
```

Deinitialize the LVGL display driver for NuttX using the provided custom configuration information.

**Parametri**

- result** -- Pointer to the `lv_nuttx_result_t` structure containing display and input device handler.

```
void lv_nuttx_run(lv_nuttx_result_t *result)
```

Call `lv_timer_handler()` (LVGL's super loop) in an endless loop. If `LV_USE_NUTTX_LIBUV` is enabled an UV timer will be created, else `lv_timer_handler()` will be called in a loop with some sleep.

**Parametri**

- result** -- pointer to a variable initialized by `lv_nuttx_init()` or `lv_nuttx_init_custom()`

```
uint32_t lv_nuttx_get_idle(void)
```

Get the idle percentage of the system.

**Ritorna**

The idle percentage of the system.

```
struct lv_nuttx_dsc_t
```

**Public Members**

```
const char *fb_path
const char *input_path
const char *utouch_path
const char *mouse_path
const char *trace_path

struct lv_nuttx_result_t
```

**Public Members**

```
lv_display_t *disp
lv_indev_t *indev
lv_indev_t *utouch_indev
lv_indev_t *mouse_indev

struct _lv_nuttx_ctx_t
```

**Public Members**

```
void *image_cache
int trace_fd
```

**lv\_nuttx\_fbdev.h****Functions**

`lv_display_t *lv_nuttx_fbdev_create(void)`

Create a new display with NuttX backend.

`int lv_nuttx_fbdev_set_file(lv_display_t *disp, const char *file)`

Initialize display with specified framebuffer device

**Parametri**

- **disp** -- pointer to display with NuttX backend
- **file** -- the name of framebuffer device

**lv\_nuttx\_image\_cache.h****Functions**

`void lv_nuttx_image_cache_init(bool use_independent_image_heap)`

`void lv_nuttx_image_cache_deinit(void)`

**lv\_nuttx\_lcd.h****Functions**

```
lv_display_t *lv_nuttx_lcd_create(const char *dev_path)
```

**lv\_nuttx\_libuv.h****Functions**

```
void *lv_nuttx_uv_init(lv_nuttx_uv_t *uv_info)
```

Initialize the uv\_loop using the provided configuration information.

**Parametri**

**uv\_info** -- Pointer to the *lv\_nuttx\_uv\_t* structure to be initialized.

```
void lv_nuttx_uv_deinit(void **data)
```

Deinitialize the uv\_loop configuration for NuttX porting layer.

**Parametri**

**data** -- Pointer to user data.

```
struct lv_nuttx_uv_t
```

**Public Members**

void \*loop

lv\_display\_t \*disp

lv\_indev\_t \*indev

**lv\_nuttx\_mouse.h****Functions**

```
lv_indev_t *lv_nuttx_mouse_create(const char *dev_path)
```

Initialize indev with specified input device.

**Parametri**

**dev\_path** -- path of input device

**lv\_nuttx\_profiler.h****Functions**

```
void lv_nuttx_profiler_init(void)
```

```
void lv_nuttx_profiler_set_file(const char *file)
```

```
void lv_nuttx_profiler_deinit(void)
```

**lv\_nuttx\_touchscreen.h****Functions**

```
lv_indev_t *lv_nuttx_touchscreen_create(const char *dev_path)
```

Initialize indev with specified input device.

**Parametri**

**dev\_path** -- path of input device

## 14.13.8 qnx

### lv\_qnx.h

LVGL driver for the QNX Screen compositing window manager.

#### Functions

`lv_display_t *lv_qnx_window_create(int32_t hor_res, int32_t ver_res)`

Create a window to use as a display for LVGL.

##### Parametri

- **hor\_res** -- The horizontal resolution (size) of the window
- **ver\_res** -- The vertical resolution (size) of the window

##### Ritorna

A pointer to a new display object if successful, NULL otherwise

`void lv_qnx_window_set_title(lv_display_t *disp, const char *title)`

Set the title of the window identified by the given display.

##### Parametri

- **disp** -- The display object for the window
- **title** -- The new title to set

`bool lv_qnx_add_pointer_device(lv_display_t *disp)`

Create a pointer input device for the display. Only one pointer object is currently supported.

##### Parametri

**disp** -- The display object associated with the device

##### Ritorna

true if successful, false otherwise

`bool lv_qnx_add_keyboard_device(lv_display_t *disp)`

Create a keyboard input device for the display. Only one keyboard object is currently supported.

##### Parametri

**disp** -- The display object associated with the device

##### Ritorna

true if successful, false otherwise

`int lv_qnx_event_loop(lv_display_t *disp)`

Runs the event loop for the display. The function only returns in response to a close event.

##### Parametri

**disp** -- The display for the event loop

##### Ritorna

Exit code

## 14.13.9 sdl

### lv\_sdl\_keyboard.h

#### Defines

`KEYBOARD_BUFFER_SIZE`

#### Functions

`lv_indev_t *lv_sdl_keyboard_create(void)`

**lv\_sdl\_mouse.h****Functions**

```
lv_indev_t *lv_sdl_mouse_create(void)
```

**lv\_sdl\_mousewheel.h****Functions**

```
lv_indev_t *lv_sdl_mousewheel_create(void)
```

**lv\_sdl\_private.h****Functions**

```
void lv_sdl_keyboard_handler(SDL_Event *event)
```

```
void lv_sdl_mouse_handler(SDL_Event *event)
```

```
void lv_sdl_mousewheel_handler(SDL_Event *event)
```

```
lv_display_t *lv_sdl_get_disp_from_win_id(uint32_t win_id)
```

**lv\_sdl\_window.h****Defines****LV SDL MOUSEWHEEL MODE ENCODER**

Modified by NXP in 2025

**LV SDL MOUSEWHEEL MODE CROWN****Functions**

```
lv_display_t *lv_sdl_window_create(int32_t hor_res, int32_t ver_res)
```

```
void lv_sdl_window_set_resizeable(lv_display_t *disp, bool value)
```

```
void lv_sdl_window_set_zoom(lv_display_t *disp, float zoom)
```

```
float lv_sdl_window_get_zoom(lv_display_t *disp)
```

```
void lv_sdl_window_set_title(lv_display_t *disp, const char *title)
```

```
void lv_sdl_window_set_icon(lv_display_t *disp, void *icon, int32_t width, int32_t height)
```

```
void *lv_sdl_window_get_renderer(lv_display_t *disp)
```

```
void lv_sdl_quit(void)
```

**14.13.10 uefi****lv\_uefi.h****lv\_uefi\_context.h****Functions**

**void `lv_uefi_init`(EFI\_HANDLE image\_handle, EFI\_SYSTEM\_TABLE \*system\_table)**

Initialize the UEFI cache variables.

**ⓘ Remark**

This has to be called before `lv_init()`.

**Parametri**

- **image\_handle** -- The handle of the current image
- **system\_table** -- Pointer to the system table

**void `lv_uefi_platform_init`(void)**

Initialize the LVGL UEFI backend.

**ⓘ Remark**

This is a private API which is used for LVGL UEFI backend implementation. LVGL users shouldn't use that because the LVGL has already used it in `lv_init`.

**void `lv_uefi_platform_deinit`(void)**

Cleanup the LVGL UEFI backend.

**ⓘ Remark**

This is a private API which is used for LVGL UEFI backend implementation. LVGL users shouldn't use that because the LVGL has already used it in `lv_deinit`.

## lv\_uefi\_display.h

### Functions

**lv\_display\_t \*`lv_uefi_display_create`(void \*handle)**

Create a LVGL display object.

**Parametri**

**handle** -- The handle on which an instance of the EFI\_GRAPHICS\_OUTPUT\_PROTOCOL protocol is installed.

**Ritorna**

The created LVGL display object.

**void \*`lv_uefi_display_get_active`(void)**

Try to find the active display handle.

**ⓘ Remark**

The active display need interfaces for EFI\_GRAPHICS\_OUTPUT\_PROTOCOL and EFI\_EDID\_ACTIVE\_PROTOCOL

**Ritorna**

The handle or NULL if not found.

**void \*`lv_uefi_display_get_any`(void)**

Try to find any display handle.

**Ritorna**

The handle or NULL if not found.

[lv\\_uefi\\_edk2.h](#)[lv\\_uefi\\_gnu\\_efi.h](#)[lv\\_uefi\\_indev.h](#)

## Functions

**lv\_indev\_t \*lv\_uefi\_simple\_pointer\_indev\_create(lv\_point\_t \*display\_res)**

Create an indev object.

**Parametri**

- display\_res** -- The resolution of the display in pixels, needed to scale the input. If NULL the resolution of the current default display will be used.

**Ritorna**

The created LVGL indev object.

**bool lv\_uefi\_simple\_pointer\_indev\_add\_handle(lv\_indev\_t \*indev, EFI\_HANDLE handle)**

Add an EFI\_SIMPLE\_POINTER\_PROTOCOL interface to the indev.

**Parametri**

- indev** -- Indev that was created with lv\_uefi\_simple\_pointer\_indev\_create.
- handle** -- The handle on which an instance of the EFI\_SIMPLE\_POINTER\_PROTOCOL protocol is installed.

**Ritorna**

True if the interface was added.

**void lv\_uefi\_simple\_pointer\_indev\_add\_all(lv\_indev\_t \*indev)**

Add all available EFI\_SIMPLE\_POINTER\_PROTOCOL interfaces to the indev.

**Parametri**

- indev** -- Indev that was created with lv\_uefi\_simple\_pointer\_indev\_create.

**lv\_indev\_t \*lv\_uefi\_absolute\_pointer\_indev\_create(lv\_point\_t \*display\_res)**

Create a LVGL indev object.

**Parametri**

- display\_res** -- The resolution of the display in pixels, needed to scale the input.

**Ritorna**

The created LVGL indev object.

**bool lv\_uefi\_absolute\_pointer\_indev\_add\_handle(lv\_indev\_t \*indev, EFI\_HANDLE handle)**

Add an EFI\_ABSOLUTE\_POINTER\_PROTOCOL interface to the indev.

**Parametri**

- indev** -- Indev that was created with lv\_uefi\_absolute\_pointer\_indev\_create.
- handle** -- The handle on which an instance of the EFI\_ABSOLUTE\_POINTER\_PROTOCOL protocol is installed.

**Ritorna**

True if the interface was added.

**void lv\_uefi\_absolute\_pointer\_indev\_add\_all(lv\_indev\_t \*indev)**

Add all available EFI\_ABSOLUTE\_POINTER\_PROTOCOL interfaces to the indev.

**Parametri**

- indev** -- Indev that was created with lv\_uefi\_absolute\_pointer\_indev\_create.

**lv\_indev\_t \*lv\_uefi\_simple\_text\_input\_indev\_create(void)**

Create an indev object.

**Ritorna**

The created LVGL indev object.

**bool lv\_uefi\_simple\_text\_input\_indev\_add\_handle(lv\_indev\_t \*indev, EFI\_HANDLE handle)**

Add an EFI\_SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL interface to the indev.

**Parametri**

- indev** -- Indev that was created with lv\_uefi\_simple\_text\_input\_indev\_create.
- handle** -- The handle on which an instance of the EFI\_SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL protocol is installed.

**Ritorna**

True if the interface was added.

**void lv\_uefi\_simple\_text\_input\_indev\_add\_all(lv\_indev\_t \*indev)**

Add all available EFI\_SIMPLE\_TEXT\_INPUT\_EX\_PROTOCOL interfaces to the indev.

**Parametri**

**indev** -- Indev that was created with lv\_uefi\_simple\_text\_input\_indev\_create.

**lv\_uefi\_private.h****Functions**

**bool lv\_uefi\_protocol\_test(EFI\_HANDLE handle, EFI\_GUID \*protocol)**

Test if a protocol is installed at a handle.

**Parametri**

- **handle** -- The handle on which the protocol might be installed.
- **protocol** -- The guid of the protocol.

**Ritorna**

TRUE if the protocol is installed, FALSE if not.

**void \*lv\_uefi\_protocol\_open(EFI\_HANDLE handle, EFI\_GUID \*protocol)**

Open a protocol.

**Parametri**

- **handle** -- The handle on which the protocol is installed.
- **protocol** -- The guid of the protocol.

**Ritorna**

A pointer to the interface, NULL if the protocol couldn't be opened.

**void lv\_uefi\_protocol\_close(EFI\_HANDLE handle, EFI\_GUID \*protocol)**

Close a protocol.

**Parametri**

- **handle** -- The handle on which the protocol is installed.
- **protocol** -- The guid of the protocol.

**size\_t lv\_uefi\_ucs2\_to\_ascii(const CHAR16 \*ucs2, char \*ascii, size\_t ascii\_len)**

Convert an UCS-2 string to an ASCII string. The string must contain only characters >= 0x20 and <= 0X7E.

**Parametri**

- **ucs2** -- The UCS-2 string.
- **ascii** -- The buffer to store the ASCII string.
- **ascii\_len** -- The size of the buffer in ASCII characters.

**Ritorna**

The number of characters written to the buffer or 0 if there was an error.

**size\_t lv\_uefi\_ascii\_to\_ucs2(const char \*ascii, CHAR16 \*ucs2, size\_t ucs2\_len)**

Convert an ASCII string to an UCS-2 string. The string must contain only characters >= 0x20 and <= 0X7E.

**Parametri**

- **ascii** -- The ASCII string.
- **ucs2** -- The buffer to store the UCS-2 string.
- **ucs2\_len** -- The size of the buffer in UCS-2 characters.

**Ritorna**

The number of bytes written to the buffer or 0 if there was an error.

**Variables**

**EFI\_HANDLE gLvEfiImageHandle**

Internal cache for the image handle (source: application entry point)

**EFI\_SYSTEM\_TABLE \*gLvEfist**

Internal cache for the system table (source: application entry point)

**EFI\_BOOT\_SERVICES \*gLvEfiBS**

Internal cache for the boot services table (source: gLvEfiST)

**EFI\_RUNTIME\_SERVICES \*gLvEfiRT**

Internal cache for the boot runtime service table (source: gLvEfiST)

**lv\_uefi\_std\_wrapper.h****14.13.11 wayland****lv\_wayland.h****Functions**

`uint32_t lv_wayland_timer_handler(void)`

Wrapper around lv\_timer\_handler

 **Nota**

Must be called in the application run loop instead of the regular lv\_timer\_handler provided by LVGL

**Ritorna**

time till it needs to be run next (in ms)

`int lv_wayland_get_fd(void)`

Retrieves the file descriptor of the wayland socket

**lv\_wayland\_private.h****Defines**

**LV\_WAYLAND\_XDG\_SHELL**

**LV\_WAYLAND\_DEFAULT\_CURSOR\_NAME**

**LVGL\_DRAW\_BUFFER\_DIV**

**DMG\_CACHE\_CAPACITY**

**TAG\_LOCAL**

**TAG\_BUFFER\_DAMAGE**

**TITLE\_BAR\_HEIGHT**

**BORDER\_SIZE**

**BUTTON\_MARGIN**

**BUTTON\_PADDING**

**BUTTON\_SIZE****LV\_WAYLAND\_CYCLE\_PERIOD****FIRST\_DECORATION****LAST\_DECORATION****NUM\_DECORATIONS****Enums**enum **object\_type***Values:*enumerator **OBJECT\_TITLEBAR**enumerator **OBJECT\_BUTTON\_CLOSE**enumerator **OBJECT\_BUTTON\_MAXIMIZE**enumerator **OBJECT\_BUTTON\_MINIMIZE**enumerator **OBJECT\_BORDER\_TOP**enumerator **OBJECT\_BORDER\_BOTTOM**enumerator **OBJECT\_BORDER\_LEFT**enumerator **OBJECT\_BORDER\_RIGHT**enumerator **OBJECT\_WINDOW****Functions**void **lv\_wayland\_init**(void)void **lv\_wayland\_deinit**(void)void **lv\_wayland\_wait\_flush\_cb**(lv\_display\_t \*disp)const struct wl\_callback\_listener \***lv\_wayland\_window\_get\_wl\_surface\_frame\_listener**(void)void **lv\_wayland\_window\_draw**(struct *window* \*window, uint32\_t width, uint32\_t height)lv\_result\_t **lv\_wayland\_window\_resize**(struct *window* \*window, int width, int height)void **lv\_wayland\_window\_destroy**(struct *window* \*window)const struct xdg\_surface\_listener \***lv\_wayland\_xdg\_shell\_get\_surface\_listener**(void)

```

const struct xdg_toplevel_listener *lv_wayland_xdg_shell_get_toplevel_listener(void)
const struct xdg_wm_base_listener *lv_wayland_xdg_shell_get_wm_base_listener(void)

lv_result_t lv_wayland_xdg_shell_set_maximized(struct window *window, bool maximized)
lv_result_t lv_wayland_xdg_shell_set_minimized(struct window *window)
lv_result_t lv_wayland_xdg_shell_set_fullscreen(struct window *window, bool fullscreen)
lv_result_t lv_wayland_xdg_shell_create_window(struct lv_wayland_context *app, struct window
                                              *window, const char *title)

lv_result_t lv_wayland_xdg_shell_destroy_window_toplevel(struct window *window)
lv_result_t lv_wayland_xdg_shell_destroy_window_surface(struct window *window)

void lv_wayland_xdg_shell_handle_pointer_event(struct lv_wayland_context *app, uint32_t
                                               serial, uint32_t button, uint32_t state)

const char *lv_wayland_xdg_shell_get_cursor_name(const struct lv_wayland_context *app)
void lv_wayland_xdg_shell_deinit(void)

void lv_wayland_shm_set_interface(shm_ctx_t *context, struct wl_registry *registry, uint32_t name,
                                  const char *interface, uint32_t version)

struct graphic_object *lv_wayland_shm_on_graphical_object_creation(shm_ctx_t *context,
                                                               struct graphic_object
                                                               *obj)

void lv_wayland_shm_on_graphical_object_destruction(shm_ctx_t *context, struct
                                                       graphic_object *obj)

lv_result_t lv_wayland_shm_set_draw_buffers(shm_ctx_t *context, lv_display_t *display)
lv_result_t lv_wayland_shm_create_draw_buffers(shm_ctx_t *context, struct window *window)
lv_result_t lv_wayland_shm_resize_window(shm_ctx_t *context, struct window *window, int32_t width,
                                         int32_t height)

lv_result_t lv_wayland_shm_is_ready(shm_ctx_t *context)
void lv_wayland_shm_delete_draw_buffers(shm_ctx_t *context, struct window *window)
void lv_wayland_shm_initalize_context(shm_ctx_t *context)
void lv_wayland_shm_deinit(shm_ctx_t *context)

void lv_wayland_shm_flush_partial_mode(lv_display_t *disp, const lv_area_t *area, unsigned char
                                       *color_p)

struct wl_cursor_theme *lv_wayland_shm_load_cursor_theme(shm_ctx_t *context)

void lv_wayland_dmabuf_set_interface(dmabuf_ctx_t *context, struct wl_registry *registry, uint32_t
                                      name, const char *interface, uint32_t version)

struct graphic_object *lv_wayland_dmabuf_on_graphical_object_creation(dmabuf_ctx_t
                                                                     *context, struct
                                                                     graphic_object
                                                                     *obj)

```

```

void lv_wayland_dmabuf_on_graphical_object_destruction(dmabuf_ctx_t *context, struct graphic_object *obj)

lv_result_t lv_wayland_dmabuf_set_draw_buffers(dmabuf_ctx_t *context, lv_display_t *display)
lv_result_t lv_wayland_dmabuf_create_draw_buffers(dmabuf_ctx_t *context, struct window *window)

lv_result_t lv_wayland_dmabuf_resize_window(dmabuf_ctx_t *context, struct window *window)
lv_result_t lv_wayland_dmabuf_is_ready(dmabuf_ctx_t *context)

void lv_wayland_dmabuf_destroy_draw_buffers(dmabuf_ctx_t *context, struct window *window)
void lv_wayland_dmabuf_initalize_context(dmabuf_ctx_t *context)
void lv_wayland_dmabuf_deinit(dmabuf_ctx_t *context)

void lv_wayland_dmabuf_flush_full_mode(lv_display_t *disp, const lv_area_t *area, unsigned char *color_p)

const struct smm_events *lv_wayland_sme_get_events(void)
const struct wl_seat_listener *lv_wayland_seat_get_listener(void)
const struct wl_keyboard_listener *lv_wayland_keyboard_get_listener(void)
const struct wl_pointer_listener *lv_wayland_pointer_get_listener(void)
const struct wl_touch_listener *lv_wayland_touch_get_listener(void)

void lv_wayland_cache_add_area(struct window *window, smm_buffer_t *buf, const lv_area_t *area)
void lv_wayland_cache_clear(struct window *window)
void lv_wayland_cache_apply_areas(struct window *window, void *dest, void *src, smm_buffer_t *src_buf)
void lv_wayland_cache_purge(struct window *window, smm_buffer_t *buf)

```

## Variables

struct *lv\_wayland\_context* **lv\_wl\_ctx**

struct **input**

### Public Members

uint32\_t **x**

uint32\_t **y**

*lv\_indev\_state\_t* **left\_button**

*lv\_indev\_state\_t* **right\_button**

*lv\_indev\_state\_t* **wheel\_button**

```

int16_t wheel_diff

struct input pointer

lv_key_t key

lv_indev_state_t state

struct input keyboard

lv_indev_touch_data_t touches[10]

uint8_t touch_event_cnt

uint8_t primary_id

struct seat

```

#### Public Members

```

struct wl_touch *wl_touch

struct wl_pointer *wl_pointer

struct wl_keyboard *wl_keyboard

struct xkb_keymap *keymap

struct xkb_state *state

struct seat xkb

struct graphic_object

```

#### Public Members

```

struct window *window

struct wl_surface *surface

struct wl_subsurface *subsurface

smm_buffer_t *pending_buffer

smm_group_t *buffer_group

```

```
struct input input

enum object_type type

int width

int height

struct dmabuf_ctx_t
```

**Public Members**

```
struct buffer *buffers

struct zwp_linux_dmabuf_v1 *handler

uint32_t format

struct shm_ctx_t
```

**Public Members**

```
lv_draw_buf_t *lv_draw_buf

struct wl_shm *handler

uint32_t format

struct lv_wayland_context
```

**Public Members**

```
struct wl_display *display

struct wl_registry *registry

struct wl_compositor *compositor

struct wl_subcompositor *subcompositor

struct wl_seat *wl_seat

struct wl_cursor_theme *cursor_theme

struct wl_surface *cursor_surface
```

*shm\_ctx\_t* **shm\_ctx**

struct xdg\_wm\_base \***xdg\_wm**

bool **opt\_disable\_decorations**

struct *xkb\_context* \***xkb\_context**

struct *seat* **seat**

struct *graphic\_object* \***touch\_obj**

struct *graphic\_object* \***pointer\_obj**

struct *graphic\_object* \***keyboard\_obj**

*lv\_ll\_t* **window\_ll**

lv\_timer\_t \***cycle\_timer**

bool **cursor\_flush\_pending**

struct pollfd **wayland\_pfd**

struct **window**

#### Public Members

lv\_display\_t \***lv\_disp**

lv\_indev\_t \***lv\_indev\_pointer**

lv\_indev\_t \***lv\_indev\_pointeraxis**

lv\_indev\_t \***lv\_indev\_touch**

lv\_indev\_t \***lv\_indev\_keyboard**

*lv\_wayland\_display\_close\_f\_t* **close\_cb**

struct *lv\_wayland\_context* \***wl\_ctx**

struct *xdg\_surface* \***xdg\_surface**

struct *xdg\_toplevel* \***xdg\_toplevel**

```
uint32_t wm_capabilities
```

```
struct graphic_object *body
```

```
lv_area_t cache[(32)]
```

```
unsigned char start
```

```
unsigned char end
```

```
unsigned size
```

```
struct window dmg_cache
```

```
int width
```

```
int height
```

```
bool resize_pending
```

```
int resize_width
```

```
int resize_height
```

```
bool flush_pending
```

```
bool shall_close
```

```
bool closed
```

```
bool maximized
```

```
bool fullscreen
```

```
uint32_t frame_counter
```

## **lv\_wayland\_smm.h**

### Defines

**SMM\_FD\_NAME**

**SMM\_POOL\_TAGS**

**SMM\_BUFFER\_TAGS**

**SMM\_GROUP\_TAGS****SMM\_POOL\_PROPERTIES**(p)**SMM\_BUFFER\_PROPERTIES**(b)**SMM\_GROUP\_PROPERTIES**(g)**SMM\_TAG**(o, n, v)

## Typedef

typedef void **smm\_pool\_t**typedef void **smm\_buffer\_t**typedef void **smm\_group\_t**

## Functions

void **smm\_init**(const struct *smm\_events* \*evs)void **smm\_setctx**(void \*ctx)void **smm\_deinit**(void)*smm\_group\_t* \***smm\_create**(void)void **smm\_resize**(*smm\_group\_t* \*grp, size\_t sz)void **smm\_destroy**(*smm\_group\_t* \*grp)*smm\_buffer\_t* \***smm\_acquire**(*smm\_group\_t* \*grp)void \***smm\_map**(*smm\_buffer\_t* \*buf)void **smm\_release**(*smm\_buffer\_t* \*buf)*smm\_buffer\_t* \***smm\_latest**(*smm\_group\_t* \*grp)*smm\_buffer\_t* \***smm\_next**(*smm\_buffer\_t* \*buf)struct **smm\_events**

### Public Members

void \***ctx**bool (\***new\_pool**)(void \*ctx, *smm\_pool\_t* \*pool)void (\***expand\_pool**)(void \*ctx, *smm\_pool\_t* \*pool)void (\***free\_pool**)(void \*ctx, *smm\_pool\_t* \*pool)bool (\***new\_buffer**)(void \*ctx, *smm\_buffer\_t* \*buf)

```
bool (*init_buffer)(void *ctx, smm_buffer_t *buf)

void (*free_buffer)(void *ctx, smm_buffer_t *buf)

struct smm_pool_properties
```

**Public Members**void \***tag**[(1)]size\_t **size**int **fd****struct smm\_buffer\_properties****Public Members**void \***tag**[(2)]*smm\_group\_t* \*const **group***smm\_pool\_t* \*const **pool**size\_t **offset****struct smm\_group\_properties****Public Members**void \***tag**[(1)]**lv\_wl\_keyboard.h****Functions**lv\_indev\_t \***lv\_wayland\_keyboard\_create**(void)lv\_indev\_t \***lv\_wayland\_get\_keyboard**(lv\_display\_t \*display)

Get keyboard input device for given LVGL display

**Parametri**    **display** -- LVGL display**Ritorna**

input device connected to keyboard, or NULL on error

**lv\_wl\_pointer.h****Functions**lv\_indev\_t \***lv\_wayland\_pointer\_create**(void)

`lv_indev_t *lv_wayland_get_pointer(lv_display_t *disp)`

Obtains the input device of the mouse pointer

#### Nota

It is used to create an input group on application start

#### Parametri

**disp** -- Reference to the LVGL display associated to the window

#### Ritorna

The input device

**lv\_wl\_pointer\_axis.h**

#### Functions

`lv_indev_t *lv_wayland_pointer_axis_create(void)`

`lv_indev_t *lv_wayland_get_pointeraxis(lv_display_t *display)`

Obtains the input device of the encoder

#### Nota

It is used to create an input group on application start

#### Parametri

**display** -- Reference to the LVGL display associated to the window

#### Ritorna

The input device

**lv\_wl\_touch.h**

#### Functions

`lv_indev_t *lv_wayland_touch_create(void)`

`lv_indev_t *lv_wayland_get_touchscreen(lv_display_t *display)`

Get touchscreen input device for given LVGL display

#### Parametri

**display** -- LVGL display

#### Ritorna

input device connected to touchscreen, or NULL on error

**lv\_wl\_window.h**

#### Typedef

`typedef bool (*lv_wayland_display_close_f_t)(lv_display_t *disp)`

#### Functions

`lv_display_t *lv_wayland_window_create(uint32_t hor_res, uint32_t ver_res, char *title, lv_wayland_display_close_f_t close_cb)`

Creates a window

#### Parametri

- **hor\_res** -- The width of the window in pixels
- **ver\_res** -- The height of the window in pixels
- **title** -- The title of the window

- **close\_cb** -- The callback that will be execute when the user closes the window

**Ritorna**

The LVGL display associated to the window

**void lv\_wayland\_window\_close(lv\_display\_t \*disp)**

Closes the window programmatically

**Parametri**

**disp** -- Reference to the LVGL display associated to the window

**bool lv\_wayland\_window\_is\_open(lv\_display\_t \*disp)**

Check if the window is open

**Parametri**

**disp** -- Reference to the LVGL display associated to the window

**Ritorna**

true: The window is open

**void lv\_wayland\_window\_setFullscreen(lv\_display\_t \*disp, bool fullscreen)**

Sets the fullscreen state of the window

**Parametri**

- **disp** -- Reference to the LVGL display associated to the window
- **fullscreen** -- If true the window enters fullscreen

**void lv\_wayland\_window\_setMaximized(lv\_display\_t \*disp, bool maximize)**

Sets the maximized state of the window

**Parametri**

- **disp** -- Reference to the LVGL display associated to the window
- **fullscreen** -- If true the window is maximized

## 14.13.12 windows

[lv\\_windows\\_context.h](#)

[Defines](#)

[CREATE\\_WAITABLE\\_TIMER\\_MANUAL\\_RESET](#)

[CREATE\\_WAITABLE\\_TIMER\\_HIGH\\_RESOLUTION](#)

[Typedef](#)

typedef struct \_*lv\_windows\_pointer\_context\_t* **lv\_windows\_pointer\_context\_t**

typedef struct \_*lv\_windows\_keypad\_queue\_item\_t* **lv\_windows\_keypad\_queue\_item\_t**

typedef struct \_*lv\_windows\_keypad\_context\_t* **lv\_windows\_keypad\_context\_t**

typedef struct \_*lv\_windows\_encoder\_context\_t* **lv\_windows\_encoder\_context\_t**

typedef struct \_*lv\_windows\_window\_context\_t* **lv\_windows\_window\_context\_t**

typedef struct \_*lv\_windows\_create\_display\_data\_t* **lv\_windows\_create\_display\_data\_t**

## Functions

`void lv_windows_platform_init(void)`

Initialize the LVGL Windows backend.

**Remark**

This is a private API which is used for LVGL Windows backend implementation. LVGL users shouldn't use that because the LVGL has already used it in lv\_init.

`lv_windows_window_context_t *lv_windows_get_window_context(HWND window_handle)`

Get the window context from specific LVGL display window.

**Remark**

This is a private API which is used for LVGL Windows backend implementation. LVGL users shouldn't use that because the maintainer doesn't promise the application binary interface compatibility for this API.

**Parametri**

`window_handle` -- The window handle of specific LVGL display window.

**Ritorna**

The window context from specific LVGL display window.

`struct _lv_windows_pointer_context_t`

**Public Members**

`lv_indev_state_t state`

`lv_point_t point`

`lv_indev_t *indev`

`struct _lv_windows_keypad_queue_item_t`

**Public Members**

`uint32_t key`

`lv_indev_state_t state`

`struct _lv_windows_keypad_context_t`

**Public Members**

`lv_ll_t queue`

`uint16_t utf16_high_surrogate`

```
uint16_t utf16_low_surrogate  
lv_indev_t *indev  
struct _lv_windows_encoder_context_t
```

#### Public Members

```
lv_indev_state_t state  
int16_t enc_diff  
lv_indev_t *indev  
struct _lv_windows_window_context_t
```

#### Public Members

```
lv_display_t *display_device_object  
lv_timer_t *display_timer_object  
int32_t window_dpi  
int32_t zoom_level  
bool allow_dpi_override  
bool simulator_mode  
bool display_resolution_changed  
lv_point_t requested_display_resolution  
HDC display_framebuffer_context_handle  
uint32_t *display_framebuffer_base  
size_t display_framebuffer_size  
lv_windows_pointer_context_t pointer  
lv_windows_keypad_context_t keypad  
lv_windows_encoder_context_t encoder  
struct _lv_windows_create_display_data_t
```

## Public Members

```
const wchar_t *title
int32_t hor_res
int32_t ver_res
int32_t zoom_level
bool allow_dpi_override
bool simulator_mode
HANDLE mutex
lv_display_t *display
```

## [lv\\_windows\\_display.h](#)

### Defines

**LV\_WINDOWS\_ZOOM\_BASE\_LEVEL**

**USER\_DEFAULT\_SCREEN\_DPI**

### Functions

lv\_display\_t \***lv\_windows\_create\_display**(const wchar\_t \*title, int32\_t hor\_res, int32\_t ver\_res, int32\_t zoom\_level, bool allow\_dpi\_override, bool simulator\_mode)

Create a LVGL display object.

#### Parametri

- **title** -- The window title of LVGL display.
- **hor\_res** -- The horizontal resolution value of LVGL display.
- **ver\_res** -- The vertical resolution value of LVGL display.
- **zoom\_level** -- The zoom level value. Base value is 100 a.k.a 100%.
- **allow\_dpi\_override** -- Allow DPI override if true, or follow the Windows DPI scaling setting dynamically.
- **simulator\_mode** -- Create simulator mode display if true, or create application mode display.

#### Ritorna

The created LVGL display object.

HWND **lv\_windows\_get\_display\_window\_handle**(lv\_display\_t \*display)

Get the window handle from specific LVGL display object.

#### Parametri

**display** -- The specific LVGL display object.

#### Ritorna

The window handle from specific LVGL display object.

int32\_t **lv\_windows\_zoom\_to\_logical**(int32\_t physical, int32\_t zoom\_level)

Get logical pixel value from physical pixel value taken account with zoom level.

**ⓘ Remark**

It uses the same calculation style as Windows OS implementation. It will be useful for integrate LVGL Windows backend to other Windows applications.

**Parametri**

- **physical** -- The physical pixel value taken account with zoom level.
- **zoom\_level** -- The zoom level value. Base value is 100 a.k.a 100%.

**Ritorna**

The logical pixel value.

`int32_t lv_windows_zoom_to_physical(int32_t logical, int32_t zoom_level)`

Get physical pixel value taken account with zoom level from logical pixel value.

 **ⓘ Remark**

It uses the same calculation style as Windows OS implementation. It will be useful for integrate LVGL Windows backend to other Windows applications.

**Parametri**

- **logical** -- The logical pixel value.
- **zoom\_level** -- The zoom level value. Base value is 100 a.k.a 100%.

**Ritorna**

The physical pixel value taken account with zoom level.

`int32_t lv_windows_dpi_to_logical(int32_t physical, int32_t dpi)`

Get logical pixel value from physical pixel value taken account with DPI scaling.

 **ⓘ Remark**

It uses the same calculation style as Windows OS implementation. It will be useful for integrate LVGL Windows backend to other Windows applications.

**Parametri**

- **physical** -- The physical pixel value taken account with DPI scaling.
- **dpi** -- The DPI scaling value. Base value is USER\_DEFAULT\_SCREEN\_DPI.

**Ritorna**

The logical pixel value.

`int32_t lv_windows_dpi_to_physical(int32_t logical, int32_t dpi)`

Get physical pixel value taken account with DPI scaling from logical pixel value.

 **ⓘ Remark**

It uses the same calculation style as Windows OS implementation. It will be useful for integrate LVGL Windows backend to other Windows applications.

**Parametri**

- **logical** -- The logical pixel value.
- **dpi** -- The DPI scaling value. Base value is USER\_DEFAULT\_SCREEN\_DPI.

**Ritorna**

The physical pixel value taken account with DPI scaling.

## lv\_windows\_input.h

### Functions

**HWND lv\_windows\_get\_indev\_window\_handle(lv\_indev\_t \*indev)**

Get the window handle from specific LVGL input device object.

**Parametri**

**indev** -- The specific LVGL input device object.

**Ritorna**

The window handle from specific LVGL input device object.

**lv\_indev\_t \*lv\_windows\_acquire\_pointer\_indev(lv\_display\_t \*display)**

Open a LVGL pointer input device object for the specific LVGL display object, or create it if the LVGL pointer input device object is not created or removed before.

**Parametri**

**display** -- The specific LVGL display object.

**Ritorna**

The LVGL pointer input device object for the specific LVGL display object.

**lv\_indev\_t \*lv\_windows\_acquire\_keypad\_indev(lv\_display\_t \*display)**

Open a LVGL keypad input device object for the specific LVGL display object, or create it if the LVGL keypad input device object is not created or removed before.

**Parametri**

**display** -- The specific LVGL display object.

**Ritorna**

The LVGL keypad input device object for the specific LVGL display object.

**lv\_indev\_t \*lv\_windows\_acquire\_encoder\_indev(lv\_display\_t \*display)**

Open a LVGL encoder input device object for the specific LVGL display object, or create it if the LVGL encoder input device object is not created or removed before.

**Parametri**

**display** -- The specific LVGL display object.

**Ritorna**

The LVGL encoder input device object for the specific LVGL display object.

## lv\_windows\_input\_private.h

### 14.13.13 x11

#### lv\_x11.h

##### Typedef

**typedef void (\*lv\_x11\_close\_cb)(void \*user\_data)**

optional window close callback function type

 **Vedi anche**

[lv\\_x11\\_window\\_set\\_close\\_cb](#)

### Functions

**void lv\_x11\_inputs\_create(lv\_display\_t \*disp, lv\_image\_dsc\_t const \*mouse\_img)**

create and add keyboard, mouse and scrollwheel objects and connect them to x11 display.

This is a convenience method handling the typical input initialisation of an X11 window:

- create keyboard (lv\_x11\_keyboard\_create)
- create mouse (with scrollwheel, lv\_x11\_mouse\_create lv\_x11\_mousewheel\_create)

**Parametri**

- **disp** -- [in] the created X11 display object from `lv_x11_window_create`
- **mouse\_img** -- [in] optional image description for the mouse cursor (NULL for no/invisible mouse cursor)

`lv_display_t *lv_x11_window_create(char const *title, int32_t hor_res, int32_t ver_res)`  
create the X11 display

The minimal initialisation for initializing the X11 display driver with keyboard/mouse support:

```
lv_display_t* disp = lv_x11_window_create("My Window Title", window_width, window_width);
lv_x11_inputs_create(disp, NULL);
```

or with mouse cursor icon:

```
lv_image_dsc_t mouse_symbol = {....};
lv_display_t* disp = lv_x11_window_create("My Window Title", window_width, window_width);
lv_x11_inputs_create(disp, &mouse_symbol);
```

#### Parametri

- **title** -- [in] title of the created X11 window
- **hor\_res** -- [in] horizontal resolution (=width) of the X11 window
- **ver\_res** -- [in] vertical resolution (=height) of the X11 window

#### Ritorna

pointer to the display object

`struct _x11_user_hdr_t`  
`#include <lv_x11.h>` Header of private display driver user data - for internal use only

#### Public Members

`struct _XDisplay *display`

X11 display object

`struct _x11_inp_data *inp_data`

input user data object

## 14.14 font

### 14.14.1 lv\_binfont\_loader.h

#### Functions

`lv_font_t *lv_binfont_create(const char *path)`

Loads a `lv_font_t` object from a binary font file

#### Parametri

**path** -- path to font file

#### Ritorna

pointer to font where to load

`void lv_binfont_destroy(lv_font_t *font)`

Frees the memory allocated by the `lv_binfont_create()` function

#### Parametri

**font** -- `lv_font_t` object created by the `lv_binfont_create` function

#### Variables

`const lv_font_class_t lv_binfont_font_class`

`struct lv_binfont_font_src_t`

**Public Members****uint32\_t font\_size**

Size of the font in pixels

**const char \*path**

Path to font file

**const void \*buffer**

Address of the font file in the memory

**uint32\_t buffer\_size**

Size of the font file buffer

**14.14.2 lv\_font.h****Defines****LV\_FONT\_DECLARE(font\_name)****Enums****enum lv\_font\_glyph\_format\_t**

The font format.

*Values:***enumerator LV\_FONT\_GLYPH\_FORMAT\_NONE**

Maybe not visible Legacy simple formats with no byte padding at end of the lines

**enumerator LV\_FONT\_GLYPH\_FORMAT\_A1**

1 bit per pixel

**enumerator LV\_FONT\_GLYPH\_FORMAT\_A2**

2 bit per pixel

**enumerator LV\_FONT\_GLYPH\_FORMAT\_A3**

3 bit per pixel

**enumerator LV\_FONT\_GLYPH\_FORMAT\_A4**

4 bit per pixel

**enumerator LV\_FONT\_GLYPH\_FORMAT\_A8**

8 bit per pixel

**enumerator LV\_FONT\_GLYPH\_FORMAT\_IMAGE**

Image format Advanced formats

**enumerator LV\_FONT\_GLYPH\_FORMAT\_VECTOR**

Vectorial format

enumerator **LV\_FONT\_GLYPH\_FORMAT\_SVG**

SVG format

enumerator **LV\_FONT\_GLYPH\_FORMAT\_CUSTOM**

Custom format

enum **lv\_font\_subpx\_t**

The bitmaps might be upscaled by 3 to achieve subpixel rendering.

*Values:*

enumerator **LV\_FONT\_SUBPX\_NONE**

enumerator **LV\_FONT\_SUBPX\_HOR**

enumerator **LV\_FONT\_SUBPX\_VER**

enumerator **LV\_FONT\_SUBPX\_BOTH**

enum **lv\_font\_kerning\_t**

Adjust letter spacing for specific character pairs.

*Values:*

enumerator **LV\_FONT\_KERNING\_NORMAL**

enumerator **LV\_FONT\_KERNING\_NONE**

## Functions

const void \***lv\_font\_get\_glyph\_bitmap**(*lv\_font\_glyph\_dsc\_t* \**g\_dsc*, lv\_draw\_buf\_t \**draw\_buf*)

Return with the bitmap of a font. It always converts the normal fonts to A8 format in a draw\_buf with LV\_DRAW\_BUF\_ALIGN and LV\_DRAW\_BUF\_STRIDE\_ALIGN

### Nota

You must call *lv\_font\_get\_glyph\_dsc()* to get *g\_dsc* (*lv\_font\_glyph\_dsc\_t*) before you can call this function.

#### Parametri

- **g\_dsc** -- the glyph descriptor including which font to use, which supply the *glyph\_index* and the format.
- **draw\_buf** -- a draw buffer that can be used to store the bitmap of the glyph.

#### Ritorna

pointer to the glyph's data. It can be a draw buffer for bitmap fonts or an image source for imgfonts.

const void \***lv\_font\_get\_glyph\_static\_bitmap**(*lv\_font\_glyph\_dsc\_t* \**g\_dsc*)

Return the bitmap as it is. It works only if the font stores the bitmap in a non-volatile memory.

#### Parametri

**g\_dsc** -- the glyph descriptor including which font to use, which supply the *glyph\_index* and the format.

**Ritorna**

the bitmap as it is

```
bool lv_font_get_glyph_dsc(const lv_font_t *font, lv_font_glyph_desc_t *dsc_out, uint32_t letter,
                           uint32_t letter_next)
```

Get the descriptor of a glyph

**Parametri**

- **font** -- pointer to font
- **dsc\_out** -- store the result descriptor here
- **letter** -- a UNICODE letter code
- **letter\_next** -- the next letter after **letter**. Used for kerning

**Ritorna**

true: descriptor is successfully loaded into **dsc\_out**. false: the letter was not found, no data is loaded to **dsc\_out**

```
void lv_font_glyph_release_draw_data(lv_font_glyph_desc_t *g_dsc)
```

Release the bitmap of a font.

 **Nota**

You must call `lv_font_get_glyph_dsc()` to get **g\_dsc** (`lv_font_glyph_desc_t`) before you can call this function.

**Parametri**

**g\_dsc** -- the glyph descriptor including which font to use, which supply the `glyph_index` and the format.

```
uint16_t lv_font_get_glyph_width(const lv_font_t *font, uint32_t letter, uint32_t letter_next)
```

Get the width of a glyph with kerning

**Parametri**

- **font** -- pointer to a font
- **letter** -- a UNICODE letter
- **letter\_next** -- the next letter after **letter**. Used for kerning

**Ritorna**

the width of the glyph

```
int32_t lv_font_get_line_height(const lv_font_t *font)
```

Get the line height of a font. All characters fit into this height

**Parametri**

**font** -- pointer to a font

**Ritorna**

the height of a font

```
void lv_font_set_kerning(lv_font_t *font, lv_font_kerning_t kerning)
```

Configure the use of kerning information stored in a font

**Parametri**

- **font** -- pointer to a font
- **kerning** -- `LV_FONT_KERNING_NORMAL` (default) or `LV_FONT_KERNING_NONE`

```
const lv_font_t*lv_font_get_default(void)
```

Get the default font, defined by `LV_FONT_DEFAULT`

**Ritorna**

return pointer to the default font

```
bool lv_font_info_is_equal(const lv_font_info_t *ft_info_1, const lv_font_info_t *ft_info_2)
```

Compare font information.

**Parametri**

- **ft\_info\_1** -- font information 1.

- **ft\_info\_2** -- font information 2.

**Ritorna**

return true if the fonts are equal.

bool **lv\_font\_has\_static\_bitmap**(const lv\_font\_t \*font)

Checks if a font has a static rendering bitmap.

**Parametri**

**font** -- pointer to a font

**Ritorna**

return true if the font has a bitmap generated for static rendering.

**Variables**

const lv\_font\_t **lv\_font\_montserrat\_8**

const lv\_font\_t **lv\_font\_montserrat\_10**

const lv\_font\_t **lv\_font\_montserrat\_12**

const lv\_font\_t **lv\_font\_montserrat\_14**

const lv\_font\_t **lv\_font\_montserrat\_16**

const lv\_font\_t **lv\_font\_montserrat\_18**

const lv\_font\_t **lv\_font\_montserrat\_20**

const lv\_font\_t **lv\_font\_montserrat\_22**

const lv\_font\_t **lv\_font\_montserrat\_24**

const lv\_font\_t **lv\_font\_montserrat\_26**

const lv\_font\_t **lv\_font\_montserrat\_28**

const lv\_font\_t **lv\_font\_montserrat\_30**

const lv\_font\_t **lv\_font\_montserrat\_32**

const lv\_font\_t **lv\_font\_montserrat\_34**

const lv\_font\_t **lv\_font\_montserrat\_36**

const lv\_font\_t **lv\_font\_montserrat\_38**

const lv\_font\_t **lv\_font\_montserrat\_40**

const lv\_font\_t **lv\_font\_montserrat\_42**

```
const lv_font_t lv_font_montserrat_44

const lv_font_t lv_font_montserrat_46

const lv_font_t lv_font_montserrat_48

const lv_font_t lv_font_montserrat_28_compressed

const lv_font_t lv_font_dejavu_16_persian_hebrew

const lv_font_t lv_font_source_han_sans_sc_14_cjk

const lv_font_t lv_font_source_han_sans_sc_16_cjk

const lv_font_t lv_font_unscii_8

const lv_font_t lv_font_unscii_16

struct lv_font_glyph_dsc_t
```

*#include <lv\_font.h>* Describes the properties of a glyph.

### Public Members

const lv\_font\_t \***resolved\_font**

Pointer to a font where the glyph was actually found after handling fallbacks

uint16\_t **adv\_w**

The glyph needs this space. Draw the next glyph after this width.

uint16\_t **box\_w**

Width of the glyph's bounding box

uint16\_t **box\_h**

Height of the glyph's bounding box

int16\_t **ofs\_x**

x offset of the bounding box

int16\_t **ofs\_y**

y offset of the bounding box

uint16\_t **stride**

Bytes in each line. If 0 than there is no padding at the end of the line.

*lv\_font\_glyph\_format\_t* **format**

Font format of the glyph see *lv\_font\_glyph\_format\_t*

**uint8\_t is\_placeholder**

Glyph is missing. But placeholder will still be displayed

**uint8\_t req\_raw\_bitmap**

0: Get bitmap should return an A8 or ARGB8888 image. 1: return the bitmap as it is (Maybe A1/2/4 or any proprietary formats).

**int32\_t outline\_stroke\_width**

used with freetype vector fonts - width of the letter border

**uint32\_t index**

Unicode code point

**const void \*src**

Pointer to the source data used by image fonts

**union *lv\_font\_glyph\_dsc\_t* gid**

The index of the glyph in the font file. Used by the font cache

**lv\_cache\_entry\_t \*entry**

The cache entry of the glyph draw data. Used by the font cache

**struct \_lv\_font\_t**

#include <lv\_font.h> Describe the properties of a font

**Public Members****bool (\*get\_glyph\_dsc)(const lv\_font\_t\*, *lv\_font\_glyph\_dsc\_t*\*, uint32\_t letter, uint32\_t letter\_next)**

Get a glyph's descriptor from a font

**const void \*(\*get\_glyph\_bitmap)(*lv\_font\_glyph\_dsc\_t*\*, lv\_draw\_buf\_t\*)**

Get a glyph's bitmap from a font

**void (\*release\_glyph)(const lv\_font\_t\*, *lv\_font\_glyph\_dsc\_t*\*)**

Release a glyph

**int32\_t line\_height**

The real line height where any text fits

**int32\_t base\_line**

Base line measured from the bottom of the line\_height

**uint8\_t subpx**

An element of *lv\_font\_subpx\_t*

**uint8\_t kerning**

An element of *lv\_font\_kerning\_t*

**uint8\_t static\_bitmap**

The font will be used as static bitmap

**int8\_t underline\_position**

Distance between the top of the underline and base line (< 0 means below the base line)

**int8\_t underline\_thickness**

Thickness of the underline

**const void \*dsc**

Store implementation specific or run\_time data or caching here

**const lv\_font\_t \*fallback**

Fallback font for missing glyph. Resolved recursively

**void \*user\_data**

Custom user data for font.

**struct \_lv\_font\_class\_t****Public Members****lv\_font\_t \*(\*create\_cb)(const lv\_font\_info\_t \*info, const void \*src)**

Font creation callback function

**void (\*delete\_cb)(lv\_font\_t \*font)**

Font deletion callback function

**void \*(\*dup\_src\_cb)(const void \*src)**

Font source duplication callback function

**void (\*free\_src\_cb)(void \*src)**

Font source free callback function

**struct \_lv\_font\_info\_t****Public Members****const char \*name**

Font name, used to distinguish different font resources

**const lv\_font\_class\_t \*class\_p**

Font backend implementation

**uint32\_t size**

Font size in pixel

**uint32\_t render\_mode**

Font rendering mode, see `lv_freetype_font_render_mode_t`

`uint32_t style`  
 Font style, see `lv_freetype_font_style_t`

`lv_font_kerning_t kerning`  
 Font kerning, see `lv_font_kerning_t`

### 14.14.3 lv\_font\_fmt\_txt.h

#### Enums

enum `lv_font_fmt_txt_cmap_type_t`

Format of font character map.

*Values:*

enumerator `LV_FONT_FMT_TXT_CMAP_FORMAT0_FULL`

enumerator `LV_FONT_FMT_TXT_CMAP_SPARSE_FULL`

enumerator `LV_FONT_FMT_TXT_CMAP_FORMAT0_TINY`

enumerator `LV_FONT_FMT_TXT_CMAP_SPARSE_TINY`

enum `lv_font_fmt_txt_bitmap_format_t`

Bitmap formats

*Values:*

enumerator `LV_FONT_FMT_TXT_PLAIN`

enumerator `LV_FONT_FMT_TXT_COMPRESSED`

enumerator `LV_FONT_FMT_TXT_COMPRESSED_NO_PREFILTER`

#### Functions

const void \*`lv_font_get_bitmap_fmt_txt`(`lv_font_glyph_dsc_t` \*`g_dsc`, `lv_draw_buf_t` \*`draw_buf`)

Used as `get_glyph_bitmap` callback in lvgl's native font format if the font is uncompressed.

##### Parametri

- `g_dsc` -- the glyph descriptor including which font to use, which supply the `glyph_index` and format.
- `draw_buf` -- a draw buffer that can be used to store the bitmap of the glyph, it's OK not to use it.

##### Ritorna

pointer to an A8 bitmap (not necessarily `bitmap_out`) or NULL if `unicode_letter` not found

bool `lv_font_get_glyph_dsc_fmt_txt`(const `lv_font_t` \*`font`, `lv_font_glyph_dsc_t` \*`dsc_out`, `uint32_t` `unicode_letter`, `uint32_t` `unicode_letter_next`)

Used as `get_glyph_dsc` callback in lvgl's native font format if the font is uncompressed.

##### Parametri

- `font` -- pointer to font
- `dsc_out` -- store the result descriptor here

- **unicode\_letter** -- a UNICODE letter code
- **unicode\_letter\_next** -- the unicode letter succeeding the letter under test

**Ritorna**

true: descriptor is successfully loaded into **dsc\_out**. false: the letter was not found, no data is loaded to **dsc\_out**

**Variables**

```
const lv_font_class_t lv_builtin_font_class
```

```
struct lv_font_fmt_txt_glyph_dsc_t
```

#include <lv\_font\_fmt\_txt.h> This describes a glyph.

**Public Members**

**uint32\_t bitmap\_index**

Start index of the bitmap. A font can be max 4 GB.

**uint32\_t adv\_w**

Draw the next glyph after this width. 28.4 format (real\_value \* 16 is stored).

**uint16\_t box\_w**

Width of the glyph's bounding box

**uint16\_t box\_h**

Height of the glyph's bounding box

**int16\_t ofs\_x**

x offset of the bounding box

**int16\_t ofs\_y**

y offset of the bounding box. Measured from the top of the line

```
struct lv_font_fmt_txt_cmap_t
```

#include <lv\_font\_fmt\_txt.h> Map codepoints to a **glyph\_dsc**s Several formats are supported to optimize memory usage See [https://github.com/lvgl/lv\\_font\\_conv/blob/master/doc/font\\_spec.md](https://github.com/lvgl/lv_font_conv/blob/master/doc/font_spec.md)

**Public Members**

**uint32\_t range\_start**

First Unicode character for this range

**uint16\_t range\_length**

Number of Unicode characters related to this range. Last Unicode character = range\_start + range\_length - 1

**uint16\_t glyph\_id\_start**

First glyph ID (array index of **glyph\_dsc**) for this range

**const uint16\_t \*unicode\_list**

```
const void *glyph_id_ofs_list
    if(type == LV_FONT_FMT_TXT_CMAP_FORMAT0...) it's uint8_t *
    LV_FONT_FMT_TXT_CMAP_SPARSE...) it's uint16_t *
```

**uint16\_t list\_length**  
Length of unicode\_list and/or glyph\_id\_ofs\_list

*lv\_font\_fmt\_txt\_cmap\_type\_t type*

Type of this character map

```
struct lv_font_fmt_txt_kern_pair_t
```

#include <lv\_font\_fmt\_txt.h> A simple mapping of kern values from pairs

#### Public Members

const void \*glyph\_ids

const int8\_t \*values

uint32\_t pair\_cnt

uint32\_t glyph\_ids\_size

0: glyph\_ids is stored as uint8\_t; 1: as uint16\_t

```
struct lv_font_fmt_txt_kern_classes_t
```

#include <lv\_font\_fmt\_txt.h> More complex but more optimal class based kern value storage

#### Public Members

const int8\_t \*class\_pair\_values

left\_class\_cnt \* right\_class\_cnt value

const uint8\_t \*left\_class\_mapping

Map the glyph\_ids to classes: index -> glyph\_id -> class\_id

const uint8\_t \*right\_class\_mapping

Map the glyph\_ids to classes: index -> glyph\_id -> class\_id

uint8\_t left\_class\_cnt

uint8\_t right\_class\_cnt

```
struct lv_font_fmt_txt_dsc_t
```

#include <lv\_font\_fmt\_txt.h> Describe store for additional data for fonts

#### Public Members

**const uint8\_t \*glyph\_bitmap**

The bitmaps of all glyphs

**const lv\_font\_fmt\_txt\_glyph\_dsc\_t \*glyph\_dsc**

Describe the glyphs

**const lv\_font\_fmt\_txt\_cmap\_t \*cmaps**

Map the glyphs to Unicode characters. Array of `lv_font_cmap_fmt_txt_t` variables

**const void \*kern\_dsc**

Store kerning values. Can be `lv_font_fmt_txt_kern_pair_t` \* or `lv_font_kern_classes_fmt_txt_t` \* depending on `kern_classes`

**uint16\_t kern\_scale**

Scale kern values in 12.4 format

**uint16\_t cmap\_num**

Number of cmap tables

**uint16\_t bpp**

Bit per pixel: 1, 2, 3, 4, 8

**uint16\_t kern\_classes**

Type of `kern_dsc`

**uint16\_t bitmap\_format**

storage format of the bitmap from `lv_font_fmt_txt_bitmap_format_t`

**uint8\_t stride**

Bytes to which each line is padded. 0: means no align and padding 1: e.g. with `bpp=4` lines are aligned to 1 byte, so there can be a 4 bits of padding 4, 8, 16, 32, 64: each line is padded to the given byte boundaries

**struct lv\_builtin\_font\_src\_t**

### Public Members

**const lv\_font\_t \*font\_p**

Pointer to built-in font

**uint32\_t size**

## 14.14.4 lv\_font\_fmt\_txt\_private.h

### Enums

**enum lv\_font\_fmt\_rle\_state\_t**

*Values:*

enumerator **RLE\_STATE\_SINGLE**  
enumerator **RLE\_STATE\_REPEATED**  
enumerator **RLE\_STATE\_COUNTER**

struct **lv\_font\_fmt\_rle\_t**

#### Public Members

  uint32\_t **rdp**  
  const uint8\_t \***in**  
  uint8\_t **bpp**  
  uint8\_t **prev\_v**  
  uint8\_t **count**  
  *lv\_font\_fmt\_rle\_state\_t* **state**

### 14.14.5 lv\_symbol\_def.h

#### Defines

**LV\_SYMBOL\_BULLET**

**LV\_SYMBOL\_AUDIO**

**LV\_SYMBOL\_VIDEO**

**LV\_SYMBOL\_LIST**

**LV\_SYMBOL\_OK**

**LV\_SYMBOL\_CLOSE**

**LV\_SYMBOL\_POWER**

**LV\_SYMBOL\_SETTINGS**

**LV\_SYMBOL\_HOME**

**LV\_SYMBOL\_DOWNLOAD**

**LV\_SYMBOL\_DRIVE**

**LV\_SYMBOL\_REFRESH**

**LV\_SYMBOL\_MUTE**

**LV\_SYMBOL\_VOLUME\_MID**

**LV\_SYMBOL\_VOLUME\_MAX**

**LV\_SYMBOL\_IMAGE**

**LV\_SYMBOL\_TINT**

**LV\_SYMBOL\_PREV**

**LV\_SYMBOL\_PLAY**

**LV\_SYMBOL\_PAUSE**

**LV\_SYMBOL\_STOP**

**LV\_SYMBOL\_NEXT**

**LV\_SYMBOL\_EJECT**

**LV\_SYMBOL\_LEFT**

**LV\_SYMBOL\_RIGHT**

**LV\_SYMBOL\_PLUS**

**LV\_SYMBOL\_MINUS**

**LV\_SYMBOL\_EYE\_OPEN**

**LV\_SYMBOL\_EYE\_CLOSE**

**LV\_SYMBOL\_WARNING**

**LV\_SYMBOL\_SHUFFLE**

**LV\_SYMBOL\_UP**

**LV\_SYMBOL\_DOWN**

LV\_SYMBOL\_LOOP

LV\_SYMBOL\_DIRECTORY

LV\_SYMBOL\_UPLOAD

LV\_SYMBOL\_CALL

LV\_SYMBOL\_CUT

LV\_SYMBOL\_COPY

LV\_SYMBOL\_SAVE

LV\_SYMBOL\_BARS

LV\_SYMBOL\_ENVELOPE

LV\_SYMBOL\_CHARGE

LV\_SYMBOL\_PASTE

LV\_SYMBOL\_BELL

LV\_SYMBOL\_KEYBOARD

LV\_SYMBOL\_GPS

LV\_SYMBOL\_FILE

LV\_SYMBOL\_WIFI

LV\_SYMBOL\_BATTERY\_FULL

LV\_SYMBOL\_BATTERY\_3

LV\_SYMBOL\_BATTERY\_2

LV\_SYMBOL\_BATTERY\_1

LV\_SYMBOL\_BATTERY\_EMPTY

LV\_SYMBOL\_USB

LV\_SYMBOL\_BLUETOOTH

**LV\_SYMBOL\_TRASH**

**LV\_SYMBOL\_EDIT**

**LV\_SYMBOL\_BACKSPACE**

**LV\_SYMBOL\_SD\_CARD**

**LV\_SYMBOL\_NEW\_LINE**

**LV\_SYMBOL\_DUMMY**

Invalid symbol at (U+F8FF). If written before a string then `lv_img` will show it as a label

## Enums

enum **lv\_str\_symbol\_id\_t**

*Values:*

enumerator **LV\_STR\_SYMBOL\_BULLET**

enumerator **LV\_STR\_SYMBOL\_AUDIO**

enumerator **LV\_STR\_SYMBOL\_VIDEO**

enumerator **LV\_STR\_SYMBOL\_LIST**

enumerator **LV\_STR\_SYMBOL\_OK**

enumerator **LV\_STR\_SYMBOL\_CLOSE**

enumerator **LV\_STR\_SYMBOL\_POWER**

enumerator **LV\_STR\_SYMBOL\_SETTINGS**

enumerator **LV\_STR\_SYMBOL\_HOME**

enumerator **LV\_STR\_SYMBOL\_DOWNLOAD**

enumerator **LV\_STR\_SYMBOL\_DRIVE**

enumerator **LV\_STR\_SYMBOL\_REFRESH**

enumerator **LV\_STR\_SYMBOL\_MUTE**

enumerator **LV\_STR\_SYMBOL\_VOLUME\_MID**

enumerator **LV\_STR\_SYMBOL\_VOLUME\_MAX**

enumerator **LV\_STR\_SYMBOL\_IMAGE**

enumerator **LV\_STR\_SYMBOL\_TINT**

enumerator **LV\_STR\_SYMBOL\_PREV**

enumerator **LV\_STR\_SYMBOL\_PLAY**

enumerator **LV\_STR\_SYMBOL\_PAUSE**

enumerator **LV\_STR\_SYMBOL\_STOP**

enumerator **LV\_STR\_SYMBOL\_NEXT**

enumerator **LV\_STR\_SYMBOL\_EJECT**

enumerator **LV\_STR\_SYMBOL\_LEFT**

enumerator **LV\_STR\_SYMBOL\_RIGHT**

enumerator **LV\_STR\_SYMBOL\_PLUS**

enumerator **LV\_STR\_SYMBOL\_MINUS**

enumerator **LV\_STR\_SYMBOL\_EYE\_OPEN**

enumerator **LV\_STR\_SYMBOL\_EYE\_CLOSE**

enumerator **LV\_STR\_SYMBOL\_WARNING**

enumerator **LV\_STR\_SYMBOL\_SHUFFLE**

enumerator **LV\_STR\_SYMBOL\_UP**

enumerator **LV\_STR\_SYMBOL\_DOWN**

enumerator **LV\_STR\_SYMBOL\_LOOP**

enumerator **LV\_STR\_SYMBOL\_DIRECTORY**

enumerator **LV\_STR\_SYMBOL\_UPLOAD**

enumerator **LV\_STR\_SYMBOL\_CALL**

enumerator **LV\_STR\_SYMBOL\_CUT**

enumerator **LV\_STR\_SYMBOL\_COPY**

enumerator **LV\_STR\_SYMBOL\_SAVE**

enumerator **LV\_STR\_SYMBOL\_BARS**

enumerator **LV\_STR\_SYMBOL\_ENVELOPE**

enumerator **LV\_STR\_SYMBOL\_CHARGE**

enumerator **LV\_STR\_SYMBOL\_PASTE**

enumerator **LV\_STR\_SYMBOL\_BELL**

enumerator **LV\_STR\_SYMBOL\_KEYBOARD**

enumerator **LV\_STR\_SYMBOL\_GPS**

enumerator **LV\_STR\_SYMBOL\_FILE**

enumerator **LV\_STR\_SYMBOL\_WIFI**

enumerator **LV\_STR\_SYMBOL\_BATTERY\_FULL**

enumerator **LV\_STR\_SYMBOL\_BATTERY\_3**

enumerator **LV\_STR\_SYMBOL\_BATTERY\_2**

enumerator **LV\_STR\_SYMBOL\_BATTERY\_1**

enumerator **LV\_STR\_SYMBOL\_BATTERY\_EMPTY**

enumerator **LV\_STR\_SYMBOL\_USB**

enumerator **LV\_STR\_SYMBOL\_BLUETOOTH**

enumerator **LV\_STR\_SYMBOL\_TRASH**

enumerator **LV\_STR\_SYMBOL\_EDIT**

enumerator **LV\_STR\_SYMBOL\_BACKSPACE**

enumerator **LV\_STR\_SYMBOL\_SD\_CARD**

enumerator **LV\_STR\_SYMBOL\_NEW\_LINE**

enumerator **LV\_STR\_SYMBOL\_DUMMY**

## 14.15 indev

### 14.15.1 lv\_indev.h

#### Typedef

typedef void (\***lv\_indev\_read\_cb\_t**)(lv\_indev\_t \*indev, *lv\_indev\_data\_t* \*data)

#### Enums

enum **lv\_indev\_type\_t**

Possible input device types

*Values:*

enumerator **LV\_INDEV\_TYPE\_NONE**

Uninitialized state

enumerator **LV\_INDEV\_TYPE\_POINTER**

Touch pad, mouse, external button

enumerator **LV\_INDEV\_TYPE\_KEYPAD**

Keypad or keyboard

enumerator **LV\_INDEV\_TYPE\_BUTTON**

External (hardware button) which is assigned to a specific point of the screen

enumerator **LV\_INDEV\_TYPE\_ENCODER**

Encoder with only Left, Right turn and a Button

enum **lv\_indev\_state\_t**

States for input devices

*Values:*

enumerator **LV\_INDEV\_STATE\_RELEASED**

enumerator **LV\_INDEV\_STATE\_PRESSED**

enum **lv\_indev\_mode\_t**

*Values:*

enumerator **LV\_INDEV\_MODE\_NONE**

enumerator **LV\_INDEV\_MODE\_TIMER**

enumerator **LV\_INDEV\_MODE\_EVENT**

enum **lv\_indev\_gesture\_type\_t**

*Values:*

enumerator **LV\_INDEV\_GESTURE\_NONE**

enumerator **LV\_INDEV\_GESTURE\_PINCH**

enumerator **LV\_INDEV\_GESTURE\_SWIPE**

enumerator **LV\_INDEV\_GESTURE\_ROTATE**

enumerator **LV\_INDEV\_GESTURE\_TWO\_FINGERS\_SWIPE**

enumerator **LV\_INDEV\_GESTURE\_SCROLL**

enumerator **LV\_INDEV\_GESTURE\_CNT**

## Functions

**lv\_indev\_t \*lv\_indev\_create(void)**

Create an indev

**Ritorna**

Pointer to the created indev or NULL when allocation failed

**void lv\_indev\_delete(lv\_indev\_t \*indev)**

Remove the provided input device. Make sure not to use the provided input device afterwards anymore.

**Parametri**

**indev** -- pointer to delete

**lv\_indev\_t \*lv\_indev\_get\_next(lv\_indev\_t \*indev)**

Get the next input device.

**Parametri**

**indev** -- pointer to the current input device. NULL to initialize.

**Ritorna**

the next input device or NULL if there are no more. Provide the first input device when the parameter is NULL

**void lv\_indev\_read(lv\_indev\_t \*indev)**

Read data from an input device.

**Parametri**

**indev** -- pointer to an input device

**void lv\_indev\_read\_timer\_cb(lv\_timer\_t \*timer)**

Called periodically to read the input devices

**Parametri**

**timer** -- pointer to a timer to read

**void lv\_indev\_enable(lv\_indev\_t \*indev, bool enable)**

Enable or disable one or all input devices (default enabled)

**Parametri**

- **indev** -- pointer to an input device or NULL to enable/disable all of them
- **enable** -- true to enable, false to disable

`lv_indev_t *lv_indev_active(void)`

Get the currently processed input device. Can be used in action functions too.

**Ritorna**

pointer to the currently processed input device or NULL if no input device processing right now

`void lv_indev_set_type(lv_indev_t *indev, lv_indev_type_t indev_type)`

Set the type of an input device

**Parametri**

- **indev** -- pointer to an input device
- **indev\_type** -- the type of the input device from `lv_indev_type_t` (`LV_INDEV_TYPE_...`)

`void lv_indev_set_read_cb(lv_indev_t *indev, lv_indev_read_cb_t read_cb)`

Set a callback function to read input device data to the indev

**Parametri**

- **indev** -- pointer to an input device
- **read\_cb** -- pointer to callback function to read input device data

`void lv_indev_set_user_data(lv_indev_t *indev, void *user_data)`

Set user data to the indev

**Parametri**

- **indev** -- pointer to an input device
- **user\_data** -- pointer to user data

`void lv_indev_set_driver_data(lv_indev_t *indev, void *driver_data)`

Set driver data to the indev

**Parametri**

- **indev** -- pointer to an input device
- **driver\_data** -- pointer to driver data

`void lv_indev_set_display(lv_indev_t *indev, struct _lv_display_t *disp)`

Assign a display to the indev

**Parametri**

- **indev** -- pointer to an input device
- **disp** -- pointer to an display

`void lv_indev_set_long_press_time(lv_indev_t *indev, uint16_t long_press_time)`

Set long press time to indev

**Parametri**

- **indev** -- pointer to input device
- **long\_press\_time** -- time long press time in ms

`void lv_indev_set_long_press_repeat_time(lv_indev_t *indev, uint16_t long_press_repeat_time)`

Set long press repeat time to indev

**Parametri**

- **indev** -- pointer to input device
- **long\_press\_repeat\_time** -- long press repeat time in ms

`void lv_indev_set_scroll_limit(lv_indev_t *indev, uint8_t scroll_limit)`

Set scroll limit to the input device

**Parametri**

- **indev** -- pointer to an input device
- **scroll\_limit** -- the number of pixels to slide before actually drag the object

`void lv_indev_set_scroll_throw(lv_indev_t *indev, uint8_t scroll_throw)`

Set scroll throw slow-down to the indev. Greater value means faster slow-down

**Parametri**

- **indev** -- pointer to an input device
- **scroll\_throw** -- the slow-down in [%]

`lv_indev_type_t lv_indev_get_type(const lv_indev_t *indev)`

Get the type of an input device

**Parametri**

- `indev` -- pointer to an input device

**Ritorna**

the type of the input device from `lv_hal_indev_type_t` (`LV_INDEV_TYPE_...`)

`lv_indev_read_cb_t lv_indev_get_read_cb(lv_indev_t *indev)`

Get the callback function to read input device data to the indev

**Parametri**

- `indev` -- pointer to an input device

**Ritorna**

Pointer to callback function to read input device data or NULL if indev is NULL

`lv_indev_state_t lv_indev_get_state(const lv_indev_t *indev)`

Get the indev state

**Parametri**

- `indev` -- pointer to an input device

**Ritorna**

Indev state or `LV_INDEV_STATE_RELEASED` if indev is NULL

`lv_group_t *lv_indev_get_group(const lv_indev_t *indev)`

Get the indev assigned group

**Parametri**

- `indev` -- pointer to an input device

**Ritorna**

Pointer to indev assigned group or NULL if indev is NULL

`lv_display_t *lv_indev_get_display(const lv_indev_t *indev)`

Get a pointer to the assigned display of the indev

**Parametri**

- `indev` -- pointer to an input device

**Ritorna**

pointer to the assigned display or NULL if indev is NULL

`void *lv_indev_get_user_data(const lv_indev_t *indev)`

Get a pointer to the user data of the indev

**Parametri**

- `indev` -- pointer to an input device

**Ritorna**

pointer to the user data or NULL if indev is NULL

`void *lv_indev_get_driver_data(const lv_indev_t *indev)`

Get a pointer to the driver data of the indev

**Parametri**

- `indev` -- pointer to an input device

**Ritorna**

pointer to the driver data or NULL if indev is NULL

`bool lv_indev_get_press_moved(const lv_indev_t *indev)`

Get whether indev is moved while pressed

**Parametri**

- `indev` -- pointer to an input device

**Ritorna**

true: indev is moved while pressed; false: indev is not moved while pressed

`void lv_indev_reset(lv_indev_t *indev, lv_obj_t *obj)`

Reset one or all input devices

**Parametri**

- `indev` -- pointer to an input device to reset or NULL to reset all of them

- **obj** -- pointer to an object which triggers the reset.

**void lv\_indev\_stop\_processing(lv\_indev\_t \*indev)**

Touch and key related events are sent to the input device first and to the widget after that. If this functions called in an indev event, the event won't be sent to the widget.

**Parametri**

**indev** -- pointer to an input device

**void lv\_indev\_reset\_long\_press(lv\_indev\_t \*indev)**

Reset the long press state of an input device

**Parametri**

**indev** -- pointer to an input device

**void lv\_indev\_set\_cursor(lv\_indev\_t \*indev, lv\_obj\_t \*cur\_obj)**

Set a cursor for a pointer input device (for LV\_INPUT\_TYPE\_POINTER and LV\_INPUT\_TYPE\_BUTTON)

**Parametri**

- **indev** -- pointer to an input device
- **cur\_obj** -- pointer to an object to be used as cursor

**void lv\_indev\_set\_group(lv\_indev\_t \*indev, lv\_group\_t \*group)**

Set a destination group for a keypad input device (for LV\_INDEV\_TYPE\_KEYPAD)

**Parametri**

- **indev** -- pointer to an input device
- **group** -- pointer to a group

**void lv\_indev\_set\_button\_points(lv\_indev\_t \*indev, const lv\_point\_t points[])**

Set the an array of points for LV\_INDEV\_TYPE\_BUTTON. These points will be assigned to the buttons to press a specific point on the screen

**Parametri**

- **indev** -- pointer to an input device
- **points** -- array of points

**void lv\_indev\_get\_point(const lv\_indev\_t \*indev, lv\_point\_t \*point)**

Get the last point of an input device (for LV\_INDEV\_TYPE\_POINTER and LV\_INDEV\_TYPE\_BUTTON)

**Parametri**

- **indev** -- pointer to an input device
- **point** -- pointer to a point to store the result

**lv\_dir\_t lv\_indev\_get\_gesture\_dir(const lv\_indev\_t \*indev)**

Get the current gesture direct

**Parametri**

**indev** -- pointer to an input device

**Ritorna**

current gesture direct

**uint32\_t lv\_indev\_get\_key(const lv\_indev\_t \*indev)**

Get the last pressed key of an input device (for LV\_INDEV\_TYPE\_KEYPAD)

**Parametri**

**indev** -- pointer to an input device

**Ritorna**

the last pressed key (0 on error)

**uint8\_t lv\_indev\_get\_short\_click\_streak(const lv\_indev\_t \*indev)**

Get the counter for consecutive clicks within a short distance and time. The counter is updated before LV\_EVENT\_SHORT\_CLICKED is fired.

**Parametri**

**indev** -- pointer to an input device

**Ritorna**

short click streak counter

`lv_dir_t lv_indev_get_scroll_dir(const lv_indev_t *indev)`

Check the current scroll direction of an input device (for LV\_INDEV\_TYPE\_POINTER and LV\_INDEV\_TYPE\_BUTTON)

**Parametri**

**indev** -- pointer to an input device

**Ritorna**

LV\_DIR\_NONE: no scrolling now LV\_DIR\_HOR/VER

`lv_obj_t *lv_indev_get_scroll_obj(const lv_indev_t *indev)`

Get the currently scrolled object (for LV\_INDEV\_TYPE\_POINTER and LV\_INDEV\_TYPE\_BUTTON)

**Parametri**

**indev** -- pointer to an input device

**Ritorna**

pointer to the currently scrolled object or NULL if no scrolling by this indev

`void lv_indev_get_vect(const lv_indev_t *indev, lv_point_t *point)`

Get the movement vector of an input device (for LV\_INDEV\_TYPE\_POINTER and LV\_INDEV\_TYPE\_BUTTON)

**Parametri**

- **indev** -- pointer to an input device
- **point** -- pointer to a point to store the types.pointer.vector

`lv_obj_t *lv_indev_get_cursor(lv_indev_t *indev)`

Get the cursor object of an input device (for LV\_INDEV\_TYPE\_POINTER only)

**Parametri**

**indev** -- pointer to an input device

**Ritorna**

pointer to the cursor object

`void lv_indev_wait_release(lv_indev_t *indev)`

Do nothing until the next release

**Parametri**

**indev** -- pointer to an input device

`lv_obj_t *lv_indev_get_active_obj(void)`

Gets a pointer to the currently active object in the currently processed input device.

**Ritorna**

pointer to currently active object or NULL if no active object

`lv_timer_t *lv_indev_get_read_timer(lv_indev_t *indev)`

Get a pointer to the indev read timer to modify its parameters with `lv_timer_...` functions.

**Parametri**

**indev** -- pointer to an input device

**Ritorna**

pointer to the indev read refresher timer. (NULL on error)

`void lv_indev_set_mode(lv_indev_t *indev, lv_indev_mode_t mode)`

Set the input device's event model: event-driven mode or timer mode.

**Parametri**

- **indev** -- pointer to an input device
- **mode** -- the mode of input device

`lv_indev_mode_t lv_indev_get_mode(lv_indev_t *indev)`

Get the input device's running mode.

**Parametri**

**indev** -- pointer to an input device

**Ritorna**

the running mode for the specified input device.

`lv_obj_t *lv_indev_search_obj(lv_obj_t *obj, lv_point_t *point)`

Search the most top, clickable object by a point

**Parametri**

- **obj** -- pointer to a start object, typically the screen
- **point** -- pointer to a point for searching the most top child

**Ritorna**

pointer to the found object or NULL if there was no suitable object

`void lv_indev_add_event_cb(lv_indev_t *indev, lv_event_cb_t event_cb, lv_event_code_t filter, void *user_data)`

Add an event handler to the indev

**Parametri**

- **indev** -- pointer to an indev
- **event\_cb** -- an event callback
- **filter** -- event code to react or LV\_EVENT\_ALL
- **user\_data** -- optional user\_data

`uint32_t lv_indev_get_event_count(lv_indev_t *indev)`

Get the number of event attached to an indev

**Parametri**

**indev** -- pointer to an indev

**Ritorna**

number of events

`lv_event_dsc_t *lv_indev_get_event_dsc(lv_indev_t *indev, uint32_t index)`

Get an event descriptor for an event

**Parametri**

- **indev** -- pointer to an indev
- **index** -- the index of the event

**Ritorna**

the event descriptor

`bool lv_indev_remove_event(lv_indev_t *indev, uint32_t index)`

Remove an event

**Parametri**

- **indev** -- pointer to an indev
- **index** -- the index of the event to remove

**Ritorna**

true: and event was removed; false: no event was removed

`uint32_t lv_indev_remove_event_cb_with_user_data(lv_indev_t *indev, lv_event_cb_t event_cb, void *user_data)`

Remove an event\_cb with user\_data

**Parametri**

- **indev** -- pointer to a indev
- **event\_cb** -- the event\_cb of the event to remove
- **user\_data** -- user\_data

**Ritorna**

the count of the event removed

`lv_result_t lv_indev_send_event(lv_indev_t *indev, lv_event_code_t code, void *param)`

Send an event to an indev

**Parametri**

- **indev** -- pointer to an indev
- **code** -- an event code. LV\_EVENT\_...
- **param** -- optional param

**Ritorna**

LV\_RESULT\_OK: indev wasn't deleted in the event.

---

```
struct lv_indev_data_t
```

#include <lv\_indev.h> Data structure passed to an input driver to fill

#### Public Members

*lv\_indev\_gesture\_type\_t* **gesture\_type**[*LV\_INDEV\_GESTURE\_CNT*]

void \***gesture\_data**[*LV\_INDEV\_GESTURE\_CNT*]

*lv\_indev\_state\_t* **state**

LV\_INDEV\_STATE\_RELEASED or LV\_INDEV\_STATE\_PRESSED

*lv\_point\_t* **point**

For LV\_INDEV\_TYPE\_POINTER the currently pressed point

uint32\_t **key**

For LV\_INDEV\_TYPE\_KEYPAD the currently pressed key

uint32\_t **btn\_id**

For LV\_INDEV\_TYPE\_BUTTON the currently pressed button

int16\_t **enc\_diff**

For LV\_INDEV\_TYPE\_ENCODER number of steps since the previous read

uint32\_t **timestamp**

Initialized to *lv\_tick\_get()*. Driver may provide more accurate timestamp for buffered events

bool **continue\_reading**

If set to true, the read callback is invoked again, unless the device is in event-driven mode

## 14.15.2 lv\_indev\_gesture.h

### Typedef

typedef struct *lv\_indev\_gesture\_recognizer* **lv\_indev\_gesture\_recognizer\_t**

typedef struct *lv\_indev\_touch\_data* **lv\_indev\_touch\_data\_t**

typedef struct *lv\_indev\_gesture* **lv\_indev\_gesture\_t**

typedef struct *lv\_indev\_gesture\_configuration* **lv\_indev\_gesture\_configuration\_t**

typedef void (\***lv\_recognizer\_func\_t**)(*lv\_indev\_gesture\_recognizer\_t\**, *lv\_indev\_touch\_data\_t\**, uint16\_t)

### Enums

enum **lv\_indev\_gesture\_state\_t**

Values:

enumerator **LV\_INDEV\_GESTURE\_STATE\_NONE**

enumerator **LV\_INDEV\_GESTURE\_STATE\_ONGOING**

enumerator **LV\_INDEV\_GESTURE\_STATE\_RECOGNIZED**

enumerator **LV\_INDEV\_GESTURE\_STATE\_ENDED**

enumerator **LV\_INDEV\_GESTURE\_STATE\_CANCELED**

## Functions

**void lv\_indev\_gesture\_detect\_pinch(*lv\_indev\_gesture\_recognizer\_t* \*recognizer,  
*lv\_indev\_touch\_data\_t* \*touches, uint16\_t touch\_cnt)**

Pinch gesture recognizer function Will update the recognizer data

### Parametri

- **recognizer** -- pointer to a gesture recognizer
- **touches** -- pointer to the first element of the collected touch events
- **touch\_cnt** -- length of passed touch event array.

**void lv\_indev\_gesture\_detect\_rotation(*lv\_indev\_gesture\_recognizer\_t* \*recognizer,  
*lv\_indev\_touch\_data\_t* \*touches, uint16\_t touch\_cnt)**

Rotation gesture recognizer function Will update the recognizer data

### Parametri

- **recognizer** -- pointer to a gesture recognizer
- **touches** -- pointer to the first element of the collected touch events
- **touch\_cnt** -- length of passed touch event array.

**void lv\_indev\_gesture\_detect\_two\_fingersSwipe(*lv\_indev\_gesture\_recognizer\_t* \*recognizer,  
*lv\_indev\_touch\_data\_t* \*touches, uint16\_t touch\_cnt)**

Two finger swipe gesture recognizer function Will update the recognizer data

### Parametri

- **recognizer** -- pointer to a gesture recognizer
- **touches** -- pointer to the first element of the collected touch events
- **touch\_cnt** -- length of passed touch event array.

**void lv\_indev\_set\_pinch\_up\_threshold(*lv\_indev\_t* \*indev, float threshold)**

Set the threshold for the pinch gesture scale up, when the scale factor of gesture reaches the threshold events get sent

### Parametri

- **indev** -- pointer to the indev device containing the pinch recognizer
- **threshold** -- threshold for a pinch up gesture to be recognized

**void lv\_indev\_set\_pinch\_down\_threshold(*lv\_indev\_t* \*indev, float threshold)**

Set the threshold for the pinch gesture scale down, when the scale factor of gesture reaches the threshold events get sent

### Parametri

- **indev** -- pointer to the indev device containing the pinch recognizer
- **threshold** -- threshold for a pinch down gesture to be recognized

**void lv\_indev\_set\_rotation\_rad\_threshold(*lv\_indev\_t* \*indev, float threshold)**

Set the rotation threshold in radian for the rotation gesture

### Parametri

- **indev** -- pointer to the indev device containing the rotation recognizer
- **threshold** -- threshold in radian for a rotation gesture to be recognized

float **lv\_event\_get\_pinch\_scale**(lv\_event\_t \*gesture\_event)

Obtains the current scale of a pinch gesture

**Parametri**

**gesture\_event** -- pointer to a gesture event

**Ritorna**

    the scale of the current gesture

float **lv\_event\_get\_rotation**(lv\_event\_t \*gesture\_event)

Obtains the current angle in radian of a rotation gesture

**Parametri**

**gesture\_event** -- pointer to a gesture event

**Ritorna**

    the rotation angle in radian of the current gesture

float **lv\_event\_get\_two\_fingersSwipeDistance**(lv\_event\_t \*gesture\_event)

Obtains the current distance in pixels of a two fingers swipe gesture, from the starting center

**Parametri**

**gesture\_event** -- pointer to a gesture event

**Ritorna**

    the distance from the center, in pixels, of the current gesture

*lv\_dir\_t* **lv\_event\_get\_two\_fingersSwipeDir**(lv\_event\_t \*gesture\_event)

Obtains the current direction from the center of a two finger swipe

**Parametri**

**gesture\_event** -- pointer to a gesture event

**Ritorna**

    the rotation angle in radian of the current gesture

void **lv\_indev\_set\_gesture\_data**(*lv\_indev\_data\_t* \*data, *lv\_indev\_gesture\_recognizer\_t* \*recognizer, *lv\_indev\_gesture\_type\_t* type)

Sets the state of the recognizer to a indev data structure, it is usually called from the indev read callback

**Parametri**

- **data** -- the indev data
- **recognizer** -- pointer to a gesture recognizer

void **lv\_indev\_get\_gesture\_center\_point**(*lv\_indev\_gesture\_recognizer\_t* \*recognizer, *lv\_point\_t* \*point)

Obtains the center point of a gesture

**Parametri**

- **gesture\_event** -- pointer to a gesture recognizer event
- **point** -- pointer to a point

*lv\_indev\_gesture\_state\_t* **lv\_event\_get\_gesture\_state**(lv\_event\_t \*gesture\_event, *lv\_indev\_gesture\_type\_t* type)

Obtains the current state of the gesture recognizer attached to an event

**Parametri**

**gesture\_event** -- pointer to a gesture recognizer event

**Ritorna**

    current state of the gesture recognizer

*lv\_indev\_gesture\_type\_t* **lv\_event\_get\_gesture\_type**(lv\_event\_t \*gesture\_event)

Obtains the current event type of the gesture recognizer attached to an event

**Parametri**

**gesture\_event** -- pointer to a gesture recognizer event

**Ritorna**

    current event type of the gesture recognizer

void **lv\_indev\_get\_gesture\_primary\_point**(*lv\_indev\_gesture\_recognizer\_t* \*recognizer, *lv\_point\_t* \*point)

Obtains the coordinates of the current primary point

**Parametri**

- **recognizer** -- pointer to a gesture recognizer
- **point** -- pointer to a point

**bool lv\_indev\_recognizer\_is\_active(*lv\_indev\_gesture\_recognizer\_t* \*recognizer)**

Allows to determine if there is an are ongoing gesture

**Parametri**

- **recognizer** -- pointer to a gesture recognizer

**Ritorna**

false if there are no contact points, or the gesture has ended - true otherwise

**void lv\_indev\_gesture\_recognizers\_update(lv\_indev\_t \*indev, *lv\_indev\_touch\_data\_t* \*touches, uint16\_t touch\_cnt)**

Update the recognizers. It execute the recognizers functions and checks for LV\_GESTURE\_STATE\_RECOGNIZED or LV\_GESTURE\_STATE\_ENDED gestures. To be called in the indev read\_cb.

**Parametri**

- **indev** -- pointer to the indev containing from which the reconizer need an update
- **touches** -- indev touch data array, containing the last touch data from indev since the last recognizers update
- **touch\_cnt** -- number of indev touch data in touches

**void lv\_indev\_gesture\_recognizers\_set\_data(lv\_indev\_t \*indev, *lv\_indev\_data\_t* \*data)**

Set the *lv\_indev\_data\_t* struct from the recognizer data. To be called in the indev read\_cb.

struct **lv\_indev\_touch\_data**

**Public Members**

*lv\_point\_t* **point**

*lv\_indev\_state\_t* **state**

uint8\_t **id**

uint32\_t **timestamp**

struct **lv\_indev\_gesture\_recognizer**

**Public Members**

*lv\_indev\_gesture\_type\_t* **type**

*lv\_indev\_gesture\_state\_t* **state**

*lv\_indev\_gesture\_t* \***info**

float **scale**

float **rotation**

float **distance**

```
float speed  
  
lv_dir_t two_fingers_swipe_dir  
  
lv_indev_gesture_configuration_t *config  
  
lv_recognizer_func_t recog_fn
```

### 14.15.3 lv\_indev\_gesture\_private.h

#### Defines

**LV\_GESTURE\_MAX\_POINTS**

#### Typedef

```
typedef struct lv_indev_gesture_motion lv_indev_gesture_motion_t
```

```
struct lv_indev_gesture_motion
```

#### Public Members

int8\_t **finger**

*lv\_point\_t* **start\_point**

*lv\_point\_t* **point**

*lv\_indev\_state\_t* **state**

```
struct lv_indev_gesture
```

#### Public Members

*lv\_indev\_gesture\_motion\_t* **motions**[2]

*lv\_point\_t* **center**

float **scale**

float **p\_scale**

float **scale\_factors\_x**[2]

float **scale\_factors\_y**[2]

```

float delta_x

float delta_y

float p_delta_x

float p_delta_y

float rotation

float p_rotation

uint8_t finger_cnt

struct lv_indev_gesture_configuration

```

**Public Members**

float **pinch\_up\_threshold**

float **pinch\_down\_threshold**

float **rotation\_angle\_rad\_threshold**

**14.15.4 lv\_indev\_private.h****Defines**

**LV\_INDEV\_VECT\_HIST\_SIZE**

**Functions**

**lv\_obj\_t \*lv\_indev\_find\_scroll\_obj(lv\_indev\_t \*indev)**

Find a scrollable object based on the current scroll vector in the indev. In handles scroll propagation to the parent if needed, and scroll directions too.

**Parametri**

**indev** -- pointer to an indev

**Ritorna**

the found scrollable object or NULL if not found.

struct **\_lv\_indev\_t**

**Public Members**

**lv\_indev\_type\_t type**

Input device type

**lv\_indev\_read\_cb\_t read\_cb**

Function pointer to read input device data.

***lv\_indev\_state\_t state***

Current state of the input device.

***lv\_indev\_state\_t prev\_state***

Previous state of the input device.

***lv\_indev\_mode\_t mode*****uint8\_t long\_pr\_sent****uint8\_t reset\_query****uint8\_t enabled****uint8\_t wait\_until\_release****uint8\_t stop\_processing\_query****uint32\_t timestamp**

Timestamp of last event

**uint32\_t pr\_timestamp**

Pressed time stamp

**uint32\_t longpr\_rep\_timestamp**

Long press repeat time stamp

**void \*driver\_data****void \*user\_data**

Pointer to the assigned display

**lv\_display\_t \*disp**

Timer to periodically read the input device

**lv\_timer\_t \*read\_timer**

Number of pixels to slide before actually drag the object

**uint8\_t scroll\_limit**

Drag throw slow-down in [%]. Greater value means faster slow-down

**uint8\_t scroll\_throw**

At least this difference should be between two points to evaluate as gesture

**uint8\_t gesture\_min\_velocity**

At least this difference should be to send a gesture

**uint8\_t gesture\_limit**

Long press time in milliseconds

**uint16\_t long\_press\_time**

Repeated trigger period in long press [ms]

**uint16\_t long\_press\_repeat\_time**

Rotary diff count will be multiplied by this value and divided by 256

**int32\_t rotary\_sensitivity*****lv\_point\_t* act\_point**

Current point of input device.

***lv\_point\_t* last\_point**

Last point of input device.

***lv\_point\_t* last\_raw\_point**

Last point read from read\_cb.

***lv\_point\_t* vect**

Difference between act\_point and last\_point.

***lv\_point\_t* vect\_hist[8]****uint32\_t vect\_hist\_timestamp[8]****uint8\_t vect\_hist\_index*****lv\_point\_t* scroll\_sum*****lv\_point\_t* scroll\_throw\_vect*****lv\_point\_t* scroll\_throw\_vect\_ori****lv\_obj\_t \*act\_obj****lv\_obj\_t \*last\_obj****lv\_obj\_t \*scroll\_obj****lv\_obj\_t \*last\_pressed****lv\_obj\_t \*last\_hovered*****lv\_area\_t* scroll\_area**

```
lv_point_t gesture_sum

int32_t diff

uint8_t short_click_streak

lv_point_t last_short_click_point

uint32_t last_short_click_timestamp

uint8_t scroll_dir

uint8_t gesture_dir

uint8_t gesture_sent

uint8_t press_moved

uint8_t pressed

struct _lv_indev_t pointer

lv_indev_state_t last_state

uint32_t last_key

struct _lv_indev_t keypad

lv_obj_t *cursor
    Cursor for LV_INPUT_TYPE_POINTER

lv_group_t *group
    Keypad destination group

const lv_point_t *btn_points
    Array points assigned to the button ()screen will be pressed here by the buttons

lv_event_list_t event_list

lv_anim_t *scroll_throw_anim

lv_indev_gesture_recognizer_t recognizers[LV_INDEV_GESTURE_CNT]

lv_indev_gesture_type_t cur_gesture

void *gesture_data[LV_INDEV_GESTURE_CNT]
```

`lv_indev_gesture_type_t gesture_type[LV_INDEV_GESTURE_CNT]`

## 14.15.5 lv\_indev\_scroll.h

### Functions

`void lv_indev_scroll_handler(lv_indev_t *indev)`

Handle scrolling. Called by LVGL during input device processing

#### Parametri

`indev` -- pointer to an input device

`void lv_indev_scroll_throw_handler(lv_indev_t *indev)`

Handle throwing after scrolling. Called by LVGL during input device processing

#### Parametri

`indev` -- pointer to an input device

`int32_t lv_indev_scroll_throw_predict(lv_indev_t *indev, lv_dir_t dir)`

Predict where would a scroll throw end

#### Parametri

- `indev` -- pointer to an input device
- `dir` -- `LV_DIR_VER` or `LV_DIR_HOR`

#### Ritorna

the difference compared to the current position when the throw would be finished

`void lv_indev_scroll_get_snap_dist(lv_obj_t *obj, lv_point_t *p)`

Get the distance of the nearest snap point

#### Parametri

- `obj` -- the object on which snap points should be found
- `p` -- save the distance of the found snap point there

## 14.16 layouts

### 14.16.1 lv\_layout.h

#### Typedef

`typedef void (*lv_layout_update_cb_t)(lv_obj_t*, void *user_data)`

#### Enums

`enum lv_layout_t`

*Values:*

enumerator `LV_LAYOUT_NONE`

enumerator `LV_LAYOUT_FLEX`

enumerator `LV_LAYOUT_GRID`

enumerator `LV_LAYOUT_LAST`

**Functions**

`uint32_t lv_layout_register(lv_layout_update_cb_t cb, void *user_data)`

Register a new layout

**Parametri**

- **cb** -- the layout update callback
- **user\_data** -- custom data that will be passed to cb

**Ritorna**

the ID of the new layout

**14.16.2 lv\_layout\_private.h****Functions**

`void lv_layout_init(void)`

`void lv_layout_deinit(void)`

`void lv_layout_apply(lv_obj_t *obj)`

Update the layout of a widget

**Parametri**

**obj** -- pointer to a widget

`struct lv_layout_dsc_t`

**Public Members**

*lv\_layout\_update\_cb\_t* **cb**

`void *user_data`

**14.16.3 flex**

`lv_flex.h`

**Defines**

`LV_FLEX_COLUMN`

`LV_FLEX_WRAP`

`LV_FLEX_REVERSE`

**Enums**

`enum lv_flex_align_t`

*Values:*

enumerator `LV_FLEX_ALIGN_START`

enumerator `LV_FLEX_ALIGN_END`

enumerator `LV_FLEX_ALIGN_CENTER`

enumerator **LV\_FLEX\_ALIGN\_SPACE\_EVENLY**

enumerator **LV\_FLEX\_ALIGN\_SPACE\_AROUND**

enumerator **LV\_FLEX\_ALIGN\_SPACE\_BETWEEN**

enum **lv\_flex\_flow\_t**  
*Values:*

- enumerator **LV\_FLEX\_FLOW\_ROW**
- enumerator **LV\_FLEX\_FLOW\_COLUMN**
- enumerator **LV\_FLEX\_FLOW\_ROW\_WRAP**
- enumerator **LV\_FLEX\_FLOW\_ROW\_REVERSE**
- enumerator **LV\_FLEX\_FLOW\_ROW\_WRAP\_REVERSE**
- enumerator **LV\_FLEX\_FLOW\_COLUMN\_WRAP**
- enumerator **LV\_FLEX\_FLOW\_COLUMN\_REVERSE**
- enumerator **LV\_FLEX\_FLOW\_COLUMN\_WRAP\_REVERSE**

## Functions

void **lv\_flex\_init**(void)

Initialize a flex layout to default values

void **lv\_obj\_set\_flex\_flow**(lv\_obj\_t \*obj, *lv\_flex\_flow\_t* flow)

Set how the item should flow

**Parametri**

- **obj** -- pointer to an object. The parent must have flex layout else nothing will happen.
- **flow** -- an element of **lv\_flex\_flow\_t**.

void **lv\_obj\_set\_flex\_align**(lv\_obj\_t \*obj, *lv\_flex\_align\_t* main\_place, *lv\_flex\_align\_t* cross\_place, *lv\_flex\_align\_t* track\_cross\_place)

Set how to place (where to align) the items and tracks

**Parametri**

- **obj** -- pointer to an object. The parent must have flex layout else nothing will happen.
- **main\_place** -- where to place the items on main axis (in their track). Any value of **lv\_flex\_align\_t**.
- **cross\_place** -- where to place the item in their track on the cross axis. **LV\_FLEX\_ALIGN\_START/END/CENTER**
- **track\_cross\_place** -- where to place the tracks in the cross direction. Any value of **lv\_flex\_align\_t**.

void **lv\_obj\_set\_flex\_grow**(lv\_obj\_t \*obj, uint8\_t grow)

Sets the width or height (on main axis) to grow the object in order fill the free space

**Parametri**

- **obj** -- pointer to an object. The parent must have flex layout else nothing will happen.

- **grow** -- a value to set how much free space to take proportionally to other growing items.

## 14.16.4 grid

**lv\_grid.h**

### Defines

**LV\_GRID\_FR(x)**

Can be used track size to make the track fill the free space.

#### Parametri

- **x** -- how much space to take proportionally to other FR tracks

#### Ritorna

a special track size

**LV\_GRID\_CONTENT**

**LV\_GRID\_TEMPLATE\_LAST**

### Enums

enum **lv\_grid\_align\_t**

*Values:*

enumerator **LV\_GRID\_ALIGN\_START**

enumerator **LV\_GRID\_ALIGN\_CENTER**

enumerator **LV\_GRID\_ALIGN\_END**

enumerator **LV\_GRID\_ALIGN\_STRETCH**

enumerator **LV\_GRID\_ALIGN\_SPACE\_EVENLY**

enumerator **LV\_GRID\_ALIGN\_SPACE\_AROUND**

enumerator **LV\_GRID\_ALIGN\_SPACE\_BETWEEN**

### Functions

void **lv\_grid\_init**(void)

void **lv\_obj\_set\_grid\_dsc\_array**(lv\_obj\_t \*obj, const int32\_t col\_dsc[], const int32\_t row\_dsc[])

void **lv\_obj\_set\_grid\_align**(lv\_obj\_t \*obj, *lv\_grid\_align\_t* column\_align, *lv\_grid\_align\_t* row\_align)

void **lv\_obj\_set\_grid\_cell**(lv\_obj\_t \*obj, *lv\_grid\_align\_t* column\_align, int32\_t col\_pos, int32\_t col\_span, *lv\_grid\_align\_t* row\_align, int32\_t row\_pos, int32\_t row\_span)

Set the cell of an object. The object's parent needs to have grid layout, else nothing will happen

#### Parametri

- **obj** -- pointer to an object
- **column\_align** -- the vertical alignment in the cell. **LV\_GRID\_START/END/CENTER/STRETCH**

- **col\_pos** -- column ID
- **col\_span** -- number of columns to take ( $\geq 1$ )
- **row\_align** -- the horizontal alignment in the cell. LV\_GRID\_START/END/CENTER/STRETCH
- **row\_pos** -- row ID
- **row\_span** -- number of rows to take ( $\geq 1$ )

```
int32_t lv_grid_fr(uint8_t x)
```

Just a wrapper to LV\_GRID\_FR for bindings.

## 14.17 libs

### 14.17.1 barcode

**lv\_barcode.h**

#### Enums

enum **lv\_barcode\_encoding\_t**

*Values:*

enumerator **LV\_BARCODE\_ENCODING\_CODE128\_GS1**

Code 128 with GS1 encoding. Strips [FCN1] and spaces.

enumerator **LV\_BARCODE\_ENCODING\_CODE128\_RAW**

Code 128 with raw encoding.

#### Functions

**lv\_obj\_t \*lv\_barcode\_create(lv\_obj\_t \*parent)**

Create an empty barcode (an **lv\_canvas**) object.

##### Parametri

- **parent** -- point to an object where to create the barcode

##### Ritorna

pointer to the created barcode object

**void lv\_barcode\_set\_dark\_color(lv\_obj\_t \*obj, lv\_color\_t color)**

Set the dark color of a barcode object

##### Parametri

- **obj** -- pointer to barcode object
- **color** -- dark color of the barcode

**void lv\_barcode\_set\_light\_color(lv\_obj\_t \*obj, lv\_color\_t color)**

Set the light color of a barcode object

##### Parametri

- **obj** -- pointer to barcode object
- **color** -- light color of the barcode

**void lv\_barcode\_set\_scale(lv\_obj\_t \*obj, uint16\_t scale)**

Set the scale of a barcode object

##### Parametri

- **obj** -- pointer to barcode object
- **scale** -- scale factor

**void lv\_barcode\_set\_direction(lv\_obj\_t \*obj, lv\_dir\_t direction)**

Set the direction of a barcode object

##### Parametri

- **obj** -- pointer to barcode object

- **direction** -- draw direction (LV\_DIR\_HOR or LB\_DIR\_VER)

**void lv\_barcode\_set\_tiled(lv\_obj\_t \*obj, bool tiled)**

Set the tiled mode of a barcode object

**Parametri**

- **obj** -- pointer to barcode object
- **tiled** -- true: tiled mode, false: normal mode (default)

**void lv\_barcode\_set\_encoding(lv\_obj\_t \*obj, lv\_barcode\_encoding\_t encoding)**

Set the encoding of a barcode object

**Parametri**

- **obj** -- pointer to barcode object
- **encoding** -- encoding (default is LV\_BARCODE\_CODE128\_GS1)

**lv\_result\_t lv\_barcode\_update(lv\_obj\_t \*obj, const char \*data)**

Set the data of a barcode object

**Parametri**

- **obj** -- pointer to barcode object
- **data** -- data to display

**Ritorna**

LV\_RESULT\_OK: if no error; LV\_RESULT\_INVALID: on error

**lv\_color\_t lv\_barcode\_get\_dark\_color(lv\_obj\_t \*obj)**

Get the dark color of a barcode object

**Parametri**

- **obj** -- pointer to barcode object

**Ritorna**

dark color of the barcode

**lv\_color\_t lv\_barcode\_get\_light\_color(lv\_obj\_t \*obj)**

Get the light color of a barcode object

**Parametri**

- **obj** -- pointer to barcode object

**Ritorna**

light color of the barcode

**uint16\_t lv\_barcode\_get\_scale(lv\_obj\_t \*obj)**

Get the scale of a barcode object

**Parametri**

- **obj** -- pointer to barcode object

**Ritorna**

scale factor

**lv\_barcode\_encoding\_t lv\_barcode\_get\_encoding(const lv\_obj\_t \*obj)**

Get the encoding of a barcode object

**Parametri**

- **obj** -- pointer to barcode object

**Ritorna**

encoding

## Variables

const lv\_obj\_class\_t **lv\_barcode\_class**

**lv\_barcode\_private.h**

struct **\_lv\_barcode\_t**

**Public Members**

`lv_canvas_t canvas`

`lv_color_t dark_color`

`lv_color_t light_color`

`uint16_t scale`

`lv_dir_t direction`

`bool tiled`

`lv_barcode_encoding_t encoding`

## 14.17.2 bin\_decoder

`lv_bin_decoder.h`

### Functions

`void lv_bin_decoder_init(void)`

Initialize the binary image decoder module

`lv_result_t lv_bin_decoder_info(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc,  
                                  lv_image_header_t *header)`

Get info about a lvgl binary image

#### Parametri

- **decoder** -- the decoder where this function belongs
- **dsc** -- image descriptor containing the source and type of the image and other info.
- **header** -- store the image data here

#### Ritorna

`LV_RESULT_OK`: the info is successfully stored in `header`; `LV_RESULT_INVALID`: unknown format or other error.

`lv_result_t lv_bin_decoder_get_area(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc,  
                                  const lv_area_t *full_area, lv_area_t *decoded_area)`

`lv_result_t lv_bin_decoder_open(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc)`

Open a lvgl binary image

#### Parametri

- **decoder** -- the decoder where this function belongs
- **dsc** -- pointer to decoder descriptor. `src`, `style` are already initialized in it.

#### Ritorna

`LV_RESULT_OK`: the info is successfully stored in `header`; `LV_RESULT_INVALID`: unknown format or other error.

`void lv_bin_decoder_close(lv_image_decoder_t *decoder, lv_image_decoder_dsc_t *dsc)`

Close the pending decoding. Free resources etc.

#### Parametri

- **decoder** -- pointer to the decoder the function associated with
- **dsc** -- pointer to decoder descriptor

### 14.17.3 bmp

**lv\_bmp.h**

#### Functions

```
void lv_bmp_init(void)
void lv_bmp_deinit(void)
```

### 14.17.4 ffmpeg

**lv\_ffmpeg.h**

#### Enums

enum **lv\_ffmpeg\_player\_cmd\_t**

*Values:*

enumerator **LV\_FFMPEG\_PLAYER\_CMD\_START**

enumerator **LV\_FFMPEG\_PLAYER\_CMD\_STOP**

enumerator **LV\_FFMPEG\_PLAYER\_CMD\_PAUSE**

enumerator **LV\_FFMPEG\_PLAYER\_CMD\_RESUME**

enumerator **LV\_FFMPEG\_PLAYER\_CMD\_LAST**

#### Functions

void **lv\_ffmpeg\_init**(void)

Register FFMPEG image decoder

void **lv\_ffmpeg\_deinit**(void)

De-initialize FFMPEG image decoder

int **lv\_ffmpeg\_get\_frame\_num**(const char \*path)

Get the number of frames contained in the file

##### Parametri

**path** -- image or video file name

##### Ritorna

Number of frames, less than 0 means failed

lv\_obj\_t \***lv\_ffmpeg\_player\_create**(lv\_obj\_t \*parent)

Create ffmpeg\_player object

##### Parametri

**parent** -- pointer to an object, it will be the parent of the new player

##### Ritorna

pointer to the created ffmpeg\_player

lv\_result\_t **lv\_ffmpeg\_player\_set\_src**(lv\_obj\_t \*obj, const char \*path)

Set the path of the file to be played.

##### Parametri

- **obj** -- pointer to a ffmpeg\_player object
- **path** -- video file path

##### Ritorna

LV\_RESULT\_OK: no error; LV\_RESULT\_INVALID: can't get the info.

---

void **lv\_ffmpeg\_player\_set\_cmd**(lv\_obj\_t \*obj, *lv\_ffmpeg\_player\_cmd\_t* cmd)

Set command control video player

**Parametri**

- **obj** -- pointer to a ffmpeg\_player object
- **cmd** -- control commands

void **lv\_ffmpeg\_player\_set\_auto\_restart**(lv\_obj\_t \*obj, bool en)

Set the video to automatically replay

**Parametri**

- **obj** -- pointer to a ffmpeg\_player object
- **en** -- true: enable the auto restart

## Variables

const lv\_obj\_class\_t **lv\_ffmpeg\_player\_class**

**lv\_ffmpeg\_private.h**

struct **\_lv\_ffmpeg\_player\_t**

### Public Members

lv\_image\_t **img**

lv\_timer\_t \***timer**

*lv\_image\_dsc\_t* **imgdsc**

bool **auto\_restart**

struct ffmpeg\_context\_s \***ffmpeg\_ctx**

## 14.17.5 freetype

**lv\_freetype.h**

### Defines

**LV\_FREETYPE\_F26DOT6\_TO\_INT(x)**

**LV\_FREETYPE\_F26DOT6\_TO\_FLOAT(x)**

**FT\_FONT\_STYLE\_NORMAL**

**FT\_FONT\_STYLE\_ITALIC**

**FT\_FONT\_STYLE\_BOLD**

### Typedef

typedef *lv\_freetype\_font\_style\_t* **LV\_FT\_FONT\_STYLE**

typedef void \***lv\_freetype\_outline\_t**

typedef const char **lv\_freetype\_font\_src\_t**

## Enums

enum **lv\_freetype\_font\_style\_t**

*Values:*

enumerator **LV\_FREETYPE\_FONT\_STYLE\_NORMAL**

enumerator **LV\_FREETYPE\_FONT\_STYLE\_ITALIC**

enumerator **LV\_FREETYPE\_FONT\_STYLE\_BOLD**

enum **lv\_freetype\_font\_render\_mode\_t**

*Values:*

enumerator **LV\_FREETYPE\_FONT\_RENDER\_MODE\_BITMAP**

enumerator **LV\_FREETYPE\_FONT\_RENDER\_MODE\_OUTLINE**

enum **lv\_freetype\_outline\_type\_t**

*Values:*

enumerator **LV\_FREETYPE\_OUTLINE\_END**

enumerator **LV\_FREETYPE\_OUTLINE\_MOVE\_TO**

enumerator **LV\_FREETYPE\_OUTLINE\_LINE\_TO**

enumerator **LV\_FREETYPE\_OUTLINE\_CUBIC\_TO**

enumerator **LV\_FREETYPE\_OUTLINE\_CONIC\_TO**

enumerator **LV\_FREETYPE\_OUTLINE\_BORDER\_START**

## Functions

lv\_result\_t **lv\_freetype\_init**(uint32\_t max\_glyph\_cnt)

Initialize the freetype library.

**Ritorna**

LV\_RESULT\_OK on success, otherwise LV\_RESULT\_INVALID.

void **lv\_freetype\_uninit**(void)

Uninitialize the freetype library

**void lv\_freetype\_init\_font\_info(lv\_font\_info\_t \*font\_info)**

Initialize a font info structure.

**Parametri**

**font\_info** -- font info structure to be initialized.

**lv\_font\_t \*lv\_freetype\_font\_create\_with\_info(const lv\_font\_info\_t \*font\_info)**

Create a freetype font with a font info structure.

**Parametri**

**font\_info** -- font info structure.

**Ritorna**

Created font, or NULL on failure.

**lv\_font\_t \*lv\_freetype\_font\_create(const char \*pathname, lv\_freetype\_font\_render\_mode\_t render\_mode, uint32\_t size, lv\_freetype\_font\_style\_t style)**

Create a freetype font.

**Parametri**

- **pathname** -- font file path.
- **render\_mode** -- font render mode(see @lv\_freetype\_font\_render\_mode\_t for details).
- **size** -- font size.
- **style** -- font style(see lv\_freetype\_font\_style\_t for details).

**Ritorna**

Created font, or NULL on failure.

**void lv\_freetype\_font\_delete(lv\_font\_t \*font)**

Delete a freetype font.

**Parametri**

**font** -- freetype font to be deleted.

**void lv\_freetype\_outline\_add\_event(lv\_event\_cb\_t event\_cb, lv\_event\_code\_t filter, void \*user\_data)**

Register a callback function to generate outlines for FreeType fonts.

**Parametri**

- **cb** -- The callback function to be registered.
- **user\_data** -- User data to be passed to the callback function.

**Ritorna**

The ID of the registered callback function, or a negative value on failure.

**uint32\_t lv\_freetype\_outline\_get\_scale(const lv\_font\_t \*font)**

Get the scale of a FreeType font.

**Parametri**

**font** -- The FreeType font to get the scale of.

**Ritorna**

The scale of the FreeType font.

**bool lv\_freetype\_is\_outline\_font(const lv\_font\_t \*font)**

Check if the font is an outline font.

**Parametri**

**font** -- The FreeType font.

**Ritorna**

Is outline font on success, otherwise false.

## Variables

const lv\_font\_class\_t **lv\_freetype\_font\_class**

**lv\_freetype\_private.h****Defines**

```
FT_ERROR_MSG(msg, error_code)

LV_FREETYPE_FONT_DSC_MAGIC_NUM
LV_FREETYPE_FONT_DSC_HAS_MAGIC_NUM(dsc)
LV_ASSERT_FREETYPE_FONT_DSC(dsc)
FT_INT_TO_F26DOT6(x)
FT_F26DOT6_TO_INT(x)
FT_INT_TO_F16DOT16(x)
FT_F16DOT16_TO_INT(x)
```

**Typedef**

```
typedef struct _lv_freetype_cache_node_t lv_freetype_cache_node_t
```

```
typedef struct _lv_freetype_context_t lv_freetype_context_t
```

```
typedef struct _lv_freetype_font_dsc_t lv_freetype_font_dsc_t
```

**Functions**

```
lv_freetype_context_t *lv_freetype_get_context(void)
```

Get the FreeType context.

**Ritorna**

A pointer to the FreeType context used by LittlevGL.

```
void lv_freetype_italic_transform(FT_Face face)
```

```
int32_t lv_freetype_italic_transform_on_pos(lv_point_t point)
```

```
lv_cache_t *lv_freetype_create_glyph_cache(uint32_t cache_size)
```

```
void lv_freetype_set_cbs_glyph(lv_freetype_font_dsc_t *dsc)
```

```
lv_cache_t *lv_freetype_create_draw_data_image(uint32_t cache_size)
```

```
void lv_freetype_set_cbs_image_font(lv_freetype_font_dsc_t *dsc)
```

```
lv_cache_t *lv_freetype_create_draw_data_outline(uint32_t cache_size)
```

```
void lv_freetype_set_cbs_outline_font(lv_freetype_font_dsc_t *dsc)
```

```
struct _lv_freetype_outline_vector_t
```

**Public Members**

```
int32_t x
```

```
int32_t y
```

```
struct lv_freetype_outline_sizes_t
```

**Public Members****int32\_t segments\_size****int32\_t data\_size****struct \_lv\_freetype\_outline\_event\_param\_t****Public Members*****lv\_freetype\_outline\_t* outline*****lv\_freetype\_outline\_type\_t* type****lv\_freetype\_outline\_vector\_t to****lv\_freetype\_outline\_vector\_t control1****lv\_freetype\_outline\_vector\_t control2*****lv\_freetype\_outline\_sizes\_t* sizes****struct \_lv\_freetype\_cache\_node\_t****Public Members****const char \*pathname*****lv\_freetype\_font\_style\_t* style*****lv\_freetype\_font\_render\_mode\_t* render\_mode****uint32\_t ref\_size**

Reference size for calculating outline glyph's real size.

**FT\_Face face*****lv\_mutex\_t* face\_lock****bool face\_has\_kerning****lv\_cache\_t \*glyph\_cache****lv\_cache\_t \*draw\_data\_cache****struct \_lv\_freetype\_context\_t**

**Public Members**

FT\_Library **library**

*lv\_ll\_t* **face\_id\_ll**

*lv\_event\_cb\_t* **event\_cb**

uint32\_t **max\_glyph\_cnt**

lv\_cache\_t \***cache\_node\_cache**

struct **\_lv\_freetype\_font\_dsc\_t**

**Public Members**

uint32\_t **magic\_num**

lv\_font\_t **font**

uint32\_t **size**

*lv\_freetype\_font\_style\_t* **style**

*lv\_freetype\_font\_render\_mode\_t* **render\_mode**

*lv\_freetype\_context\_t* \***context**

*lv\_freetype\_cache\_node\_t* \***cache\_node**

lv\_cache\_entry\_t \***cache\_node\_entry**

FTC\_FaceID **face\_id**

uint32\_t **outline\_stroke\_width**

*lv\_font\_kerning\_t* **kerning**

**14.17.6 fsdrv**

**lv\_fsdrv.h**

**Defines**

**LV\_FS\_MAX\_PATH\_LEN**

**Functions**

```
void lv_fs_fatfs_init(void)
void lv_fs_stdio_init(void)
void lv_fs_posix_init(void)
void lv_fs_win32_init(void)
void lv_fs_memfs_init(void)
void lv_littlefs_set_handler(struct lfs*)
void lv_fs_littlefs_init(void)
void lv_fs_arduino_esp_littlefs_init(void)
void lv_fs_arduino_sd_init(void)
void lv_fs_uefi_init(void)
```

**14.17.7 gif****lv\_gif.h****Functions**

`lv_obj_t *lv_gif_create(lv_obj_t *parent)`

Create a gif object.

**Parametri**

**parent** -- pointer to an object, it will be the parent of the new gif.

**Ritorna**

pointer to the gif obj

`void lv_gif_set_src(lv_obj_t *obj, const void *src)`

Set the gif data to display on the object.

**Parametri**

- **obj** -- pointer to a gif object
- **src** -- 1) pointer to an *lv\_image\_dsc\_t* descriptor (which contains gif raw data) or  
2) path to a gif file (e.g. "S:/dir/anim.gif")

`void lv_gif_restart(lv_obj_t *obj)`

Restart a gif animation.

**Parametri**

**obj** -- pointer to a gif obj

`void lv_gif_pause(lv_obj_t *obj)`

Pause a gif animation.

**Parametri**

**obj** -- pointer to a gif obj

`void lv_gif_resume(lv_obj_t *obj)`

Resume a gif animation.

**Parametri**

**obj** -- pointer to a gif obj

`bool lv_gif_is_loaded(lv_obj_t *obj)`

Checks if the GIF was loaded correctly.

**Parametri**

**obj** -- pointer to a gif obj

`int32_t lv_gif_get_loop_count(lv_obj_t *obj)`

Get the loop count for the GIF.

**Parametri**

- **obj** -- pointer to a gif obj

`void lv_gif_set_loop_count(lv_obj_t *obj, int32_t count)`

Set the loop count for the GIF.

**Parametri**

- **obj** -- pointer to a gif obj
- **count** -- the loop count to set

## Variables

`const lv_obj_class_t lv_gif_class`

**lv\_gif\_private.h**

`struct _lv_gif_t`

### Public Members

`lv_image_t img`

`gd_GIF *gif`

`lv_timer_t *timer`

`lv_image_dsc_t imgdsc`

`uint32_t last_call`

## 14.17.8 libjpeg\_turbo

**lv\_libjpeg\_turbo.h**

### Functions

`void lv_libjpeg_turbo_init(void)`

Register the JPEG-Turbo decoder functions in LVGL

`void lv_libjpeg_turbo_deinit(void)`

## 14.17.9 libpng

**lv\_libpng.h**

### Functions

`void lv_libpng_init(void)`

Register the PNG decoder functions in LVGL

`void lv_libpng_deinit(void)`

## 14.17.10 lodepng

**lv\_lodepng.h**

### Functions

**void lv\_lodepng\_init(void)**

Register the PNG decoder functions in LVGL

**void lv\_lodepng\_deinit(void)**

## 14.17.11 qrcode

**lv\_qrcode.h**

### Functions

**lv\_obj\_t \*lv\_qrcode\_create(lv\_obj\_t \*parent)**

Create an empty QR code (an **lv\_canvas**) object.

#### Parametri

- **parent** -- point to an object where to create the QR code

#### Ritorna

pointer to the created QR code object

**void lv\_qrcode\_set\_size(lv\_obj\_t \*obj, int32\_t size)**

Set QR code size.

#### Parametri

- **obj** -- pointer to a QR code object
- **size** -- width and height of the QR code

**void lv\_qrcode\_set\_dark\_color(lv\_obj\_t \*obj, lv\_color\_t color)**

Set QR code dark color.

#### Parametri

- **obj** -- pointer to a QR code object
- **color** -- dark color of the QR code

**void lv\_qrcode\_set\_light\_color(lv\_obj\_t \*obj, lv\_color\_t color)**

Set QR code light color.

#### Parametri

- **obj** -- pointer to a QR code object
- **color** -- light color of the QR code

**lv\_result\_t lv\_qrcode\_update(lv\_obj\_t \*obj, const void \*data, uint32\_t data\_len)**

Set the data of a QR code object

#### Parametri

- **obj** -- pointer to a QR code object
- **data** -- data to display
- **data\_len** -- length of data in bytes

#### Ritorna

LV\_RESULT\_OK: if no error; LV\_RESULT\_INVALID: on error

### Variables

const lv\_obj\_class\_t **lv\_qrcode\_class**

**lv\_qrcode\_private.h**

struct **\_lv\_qrcode\_t**

**Public Members**

`lv_canvas_t canvas`

`lv_color_t dark_color`

`lv_color_t light_color`

**14.17.12 rle**

`lv_rle.h`

**Functions**

`uint32_t lv_rle_decompress(const uint8_t *input, uint32_t input_buff_len, uint8_t *output, uint32_t output_buff_len, uint8_t blk_size)`

**14.17.13 rlottie**

`lv_rlottie.h`

**Enums**

enum `lv_rlottie_ctrl_t`

*Values:*

enumerator `LV_RLOTTIE_CTRL_FORWARD`

enumerator `LV_RLOTTIE_CTRL_BACKWARD`

enumerator `LV_RLOTTIE_CTRL_PAUSE`

enumerator `LV_RLOTTIE_CTRL_PLAY`

enumerator `LV_RLOTTIE_CTRL_LOOP`

**Functions**

`lv_obj_t *lv_rlottie_create_from_file(lv_obj_t *parent, int32_t width, int32_t height, const char *path)`

`lv_obj_t *lv_rlottie_create_from_raw(lv_obj_t *parent, int32_t width, int32_t height, const char *rlottie_desc)`

`void lv_rlottie_set_play_mode(lv_obj_t *rlottie, const lv_rlottie_ctrl_t ctrl)`

`void lv_rlottie_set_current_frame(lv_obj_t *rlottie, const size_t goto_frame)`

**Variables**

const `lv_obj_class_t lv_rlottie_class`

**lv\_rlottie\_private.h**

```
struct _lv_rlottie_t
```

**Public Members**

`lv_image_t img_ext`

`struct Lottie_Animation_S *animation`

`lv_timer_t *task`

`lv_image_dsc_t imgdsc`

`size_t total_frames`

`size_t current_frame`

`size_t framerate`

`uint32_t *allocated_buf`

`size_t allocated_buffer_size`

`size_t scanline_width`

`lv_rlottie_ctrl_t play_ctrl`

`size_t dest_frame`

**14.17.14 svg****lv\_svg.h****Defines**

`LV_SVG_NODE_CHILD(n, i)`

`LV_SVG_NODE(n)`

**Typedef**

`typedef int8_t lv_svg_tag_t`

`typedef uint8_t lv_svg_attr_type_t`

`typedef uint8_t lv_svg_transform_type_t`

`typedef uint32_t lv_svg_aspect_ratio_t`

```
typedef uint32_t lv_svg_color_t  
  
typedef uint8_t lv_svg_fill_rule_t  
  
typedef uint8_t lv_svg_line_cap_t  
  
typedef uint8_t lv_svg_line_join_t  
  
typedef uint8_t lv_svg_gradient_units_t  
  
typedef uint8_t lv_svg_attr_value_type_t  
  
typedef uint8_t lv_svg_attr_value_class_t
```

## Enums

enum **lv\_svg\_tag\_t**

*Values:*

enumerator **LV\_SVG\_TAG\_INVALID**

enumerator **LV\_SVG\_TAG\_CONTENT**

enumerator **LV\_SVG\_TAG\_SVG**

enumerator **LV\_SVG\_TAG\_USE**

enumerator **LV\_SVG\_TAG\_G**

enumerator **LV\_SVG\_TAG\_PATH**

enumerator **LV\_SVG\_TAG\_RECT**

enumerator **LV\_SVG\_TAG\_CIRCLE**

enumerator **LV\_SVG\_TAG\_ELLIPSE**

enumerator **LV\_SVG\_TAG\_LINE**

enumerator **LV\_SVG\_TAG\_POLYLINE**

enumerator **LV\_SVG\_TAG\_POLYGON**

enumerator **LV\_SVG\_TAG\_SOLID\_COLOR**

enumerator **LV\_SVG\_TAG\_LINEAR\_GRADIENT**

enumerator **LV\_SVG\_TAG\_RADIAL\_GRADIENT**

enumerator **LV\_SVG\_TAG\_STOP**

enumerator **LV\_SVG\_TAG\_DEFS**

enumerator **LV\_SVG\_TAG\_IMAGE**

enumerator **LV\_SVG\_TAG\_MPATH**

enumerator **LV\_SVG\_TAG\_SET**

enumerator **LV\_SVG\_TAG\_ANIMATE**

enumerator **LV\_SVG\_TAG\_ANIMATE\_COLOR**

enumerator **LV\_SVG\_TAG\_ANIMATE\_TRANSFORM**

enumerator **LV\_SVG\_TAG\_ANIMATE\_MOTION**

enumerator **LV\_SVG\_TAG\_TEXT**

enumerator **LV\_SVG\_TAG\_TSPAN**

enumerator **LV\_SVG\_TAG\_TEXT\_AREA**

enum **lv\_svg\_attr\_type\_t**

*Values:*

enumerator **LV\_SVG\_ATTR\_INVALID**

enumerator **LV\_SVG\_ATTR\_ID**

enumerator **LV\_SVG\_ATTR\_XML\_ID**

enumerator **LV\_SVG\_ATTR\_VERSION**

enumerator **LV\_SVG\_ATTR\_BASE\_PROFILE**

enumerator **LV\_SVG\_ATTR\_VIEWBOX**

enumerator **LV\_SVG\_ATTR\_PRESERVE\_ASPECT\_RATIO**

enumerator **LV\_SVG\_ATTR\_VIEWPORT\_FILL**

enumerator **LV\_SVG\_ATTR\_VIEWPORT\_FILL\_OPACITY**

enumerator **LV\_SVG\_ATTR\_DISPLAY**

enumerator **LV\_SVG\_ATTR\_VISIBILITY**

enumerator **LV\_SVG\_ATTR\_X**

enumerator **LV\_SVG\_ATTR\_Y**

enumerator **LV\_SVG\_ATTR\_WIDTH**

enumerator **LV\_SVG\_ATTR\_HEIGHT**

enumerator **LV\_SVG\_ATTR\_RX**

enumerator **LV\_SVG\_ATTR\_RY**

enumerator **LV\_SVG\_ATTR\_CX**

enumerator **LV\_SVG\_ATTR\_CY**

enumerator **LV\_SVG\_ATTR\_R**

enumerator **LV\_SVG\_ATTR\_X1**

enumerator **LV\_SVG\_ATTR\_Y1**

enumerator **LV\_SVG\_ATTR\_X2**

enumerator **LV\_SVG\_ATTR\_Y2**

enumerator **LV\_SVG\_ATTR\_POINTS**

enumerator **LV\_SVG\_ATTR\_D**

enumerator **LV\_SVG\_ATTR\_PATH\_LENGTH**

enumerator **LV\_SVG\_ATTR\_XLINK\_HREF**

enumerator **LV\_SVG\_ATTR\_FILL**

enumerator **LV\_SVG\_ATTR\_FILL\_RULE**

enumerator **LV\_SVG\_ATTR\_FILL\_OPACITY**

enumerator **LV\_SVG\_ATTR\_STROKE**

enumerator **LV\_SVG\_ATTR\_STROKE\_WIDTH**

enumerator **LV\_SVG\_ATTR\_STROKE\_LINECAP**

enumerator **LV\_SVG\_ATTR\_STROKE\_LINEJOIN**

enumerator **LV\_SVG\_ATTR\_STROKE\_MITER\_LIMIT**

enumerator **LV\_SVG\_ATTR\_STROKE\_DASH\_ARRAY**

enumerator **LV\_SVG\_ATTR\_STROKE\_DASH\_OFFSET**

enumerator **LV\_SVG\_ATTR\_STROKE\_OPACITY**

enumerator **LV\_SVG\_ATTR\_OPACITY**

enumerator **LV\_SVG\_ATTR\_SOLID\_COLOR**

enumerator **LV\_SVG\_ATTR\_SOLID\_OPACITY**

enumerator **LV\_SVG\_ATTR\_GRADIENT\_UNITS**

enumerator **LV\_SVG\_ATTR\_GRADIENT\_STOP\_OFFSET**

enumerator **LV\_SVG\_ATTR\_GRADIENT\_STOP\_COLOR**

enumerator **LV\_SVG\_ATTR\_GRADIENT\_STOP\_OPACITY**

enumerator **LV\_SVG\_ATTR\_FONT\_FAMILY**

enumerator **LV\_SVG\_ATTR\_FONT\_STYLE**

enumerator **LV\_SVG\_ATTR\_FONT\_VARIANT**

enumerator **LV\_SVG\_ATTR\_FONT\_WEIGHT**

enumerator **LV\_SVG\_ATTR\_FONT\_SIZE**

enumerator **LV\_SVG\_ATTR\_TRANSFORM**

enumerator **LV\_SVG\_ATTR\_TEXT\_ANCHOR**

enumerator **LV\_SVG\_ATTR\_ATTRIBUTE\_NAME**

enumerator **LV\_SVG\_ATTR\_ATTRIBUTE\_TYPE**

enumerator **LV\_SVG\_ATTR\_BEGIN**

enumerator **LV\_SVG\_ATTR\_END**

enumerator **LV\_SVG\_ATTR\_DUR**

enumerator **LV\_SVG\_ATTR\_MIN**

enumerator **LV\_SVG\_ATTR\_MAX**

enumerator **LV\_SVG\_ATTR\_RESTART**

enumerator **LV\_SVG\_ATTR\_REPEAT\_COUNT**

enumerator **LV\_SVG\_ATTR\_REPEAT\_DUR**

enumerator **LV\_SVG\_ATTR\_CALC\_MODE**

enumerator **LV\_SVG\_ATTR\_VALUES**

enumerator **LV\_SVG\_ATTR\_KEY\_TIMES**

enumerator **LV\_SVG\_ATTR\_KEY SPLINES**

enumerator **LV\_SVG\_ATTR\_KEY\_POINTS**

enumerator **LV\_SVG\_ATTR\_FROM**

enumerator **LV\_SVG\_ATTR\_TO**

enumerator **LV\_SVG\_ATTR\_BY**

enumerator **LV\_SVG\_ATTR\_ADDITIVE**

enumerator **LV\_SVG\_ATTR\_ACCUMULATE**

enumerator **LV\_SVG\_ATTR\_PATH**

enumerator **LV\_SVG\_ATTR\_ROTATE**

enumerator **LV\_SVG\_ATTR\_TRANSFORM\_TYPE**

enum **\_lv\_svg\_transform\_type\_t**

*Values:*

enumerator **LV\_SVG\_TRANSFORM\_TYPE\_MATRIX**

enumerator **LV\_SVG\_TRANSFORM\_TYPE\_TRANSLATE**

enumerator **LV\_SVG\_TRANSFORM\_TYPE\_ROTATE**

enumerator **LV\_SVG\_TRANSFORM\_TYPE\_SCALE**

enumerator **LV\_SVG\_TRANSFORM\_TYPE\_SKW\_X**

enumerator **LV\_SVG\_TRANSFORM\_TYPE\_SKW\_Y**

enum **lv\_svg\_anim\_action\_t**

*Values:*

enumerator **LV\_SVG\_ANIM\_REMOVE**

enumerator **LV\_SVG\_ANIM\_FREEZE**

enum **\_lv\_svg\_anim\_restart\_type\_t**

*Values:*

enumerator **LV\_SVG\_ANIM\_RESTART\_ALWAYS**

enumerator **LV\_SVG\_ANIM\_RESTART\_WHEN\_NOT\_ACTIVE**

enumerator **LV\_SVG\_ANIM\_RESTART\_NEVER**

enum **\_lv\_svg\_anim\_calc\_mode\_t**

*Values:*

enumerator **LV\_SVG\_ANIM\_CALC\_MODE\_LINEAR**

enumerator **LV\_SVG\_ANIM\_CALC\_MODE\_PACED**

enumerator **LV\_SVG\_ANIM\_CALC\_MODE\_SPLINE**

enumerator **LV\_SVG\_ANIM\_CALC\_MODE\_DISCRETE**

enum **\_lv\_svg\_anim\_additive\_type\_t**

*Values:*

enumerator **LV\_SVG\_ANIM\_ADDITIVE\_REPLACE**

enumerator **LV\_SVG\_ANIM\_ADDITIVE\_SUM**

enum **\_lv\_svg\_anim\_accumulate\_type\_t**

*Values:*

enumerator **LV\_SVG\_ANIM\_ACCUMULATE\_NONE**

enumerator **LV\_SVG\_ANIM\_ACCUMULATE\_SUM**

enum **\_lv\_svg\_aspect\_ratio\_t**

*Values:*

enumerator **LV\_SVG\_ASPECT\_RATIO\_NONE**

enumerator **LV\_SVG\_ASPECT\_RATIO\_XMIN\_YMIN**

enumerator **LV\_SVG\_ASPECT\_RATIO\_XMID\_YMIN**

enumerator **LV\_SVG\_ASPECT\_RATIO\_XMAX\_YMIN**

enumerator **LV\_SVG\_ASPECT\_RATIO\_XMIN\_YMID**

enumerator **LV\_SVG\_ASPECT\_RATIO\_XMID\_YMID**

enumerator **LV\_SVG\_ASPECT\_RATIO\_XMAX\_YMID**

enumerator **LV\_SVG\_ASPECT\_RATIO\_XMIN\_YMAX**

enumerator **LV\_SVG\_ASPECT\_RATIO\_XMID\_YMAX**

enumerator **LV\_SVG\_ASPECT\_RATIO\_XMAX\_YMAX**

enum **\_lv\_svg\_aspect\_ratio\_opt\_t**

*Values:*

enumerator **LV\_SVG\_ASPECT\_RATIO\_OPT\_MEET**

enumerator **LV\_SVG\_ASPECT\_RATIO\_OPT\_SLICE**

enum **\_lv\_svg\_fill\_rule\_t**

*Values:*

enumerator **LV\_SVG\_FILL\_NONZERO**

enumerator **LV\_SVG\_FILL\_EVENODD**

enum **\_lv\_svg\_line\_cap\_t**

*Values:*

enumerator **LV\_SVG\_LINE\_CAP\_BUTT**

enumerator **LV\_SVG\_LINE\_CAP\_SQUARE**

enumerator **LV\_SVG\_LINE\_CAP\_ROUND**

enum **\_lv\_svg\_line\_join\_t**

*Values:*

enumerator **LV\_SVG\_LINE\_JOIN\_MITER**

enumerator **LV\_SVG\_LINE\_JOIN\_BEVEL**

enumerator **LV\_SVG\_LINE\_JOIN\_ROUND**

enum **\_lv\_svg\_gradient\_units\_t**

*Values:*

enumerator **LV\_SVG\_GRADIENT\_UNITS\_OBJECT**

enumerator **LV\_SVG\_GRADIENT\_UNITS\_USER\_SPACE**

enum **\_lv\_svg\_path\_cmd\_t**

*Values:*

enumerator **LV\_SVG\_PATH\_CMD\_MOVE\_TO**

enumerator **LV\_SVG\_PATH\_CMD\_LINE\_TO**

enumerator **LV\_SVG\_PATH\_CMD\_CURVE\_TO**

enumerator **LV\_SVG\_PATH\_CMD\_QUAD\_TO**

enumerator **LV\_SVG\_PATH\_CMD\_ARC\_TO**

enumerator **LV\_SVG\_PATH\_CMD\_CLOSE**

enum **\_lv\_svg\_attr\_value\_type\_t**

*Values:*

enumerator **LV\_SVG\_ATTR\_VALUE\_DATA**

enumerator **LV\_SVG\_ATTR\_VALUE\_PTR**

---

enum **lv\_svg\_attr\_value\_class\_t**

*Values:*

enumerator **LV\_SVG\_ATTR\_VALUE\_NONE**

enumerator **LV\_SVG\_ATTR\_VALUE\_INITIAL**

enumerator **LV\_SVG\_ATTR\_VALUE\_INHERIT**

## Functions

*lv\_svg\_node\_t* \***lv\_svg\_load\_data**(const char \*svg\_data, uint32\_t data\_len)

Loading SVG data and creating the DOM tree.

**Parametri**

- **svg\_data** -- pointer to the SVG data
- **data\_len** -- the SVG data length

*lv\_svg\_node\_t* \***lv\_svg\_node\_create**(*lv\_svg\_node\_t* \*parent)

Create an SVG DOM node.

**Parametri**

- **parent** -- pointer to the parent node

**Ritorna**

true: an new SVG DOM node, false: NULL

void **lv\_svg\_node\_delete**(*lv\_svg\_node\_t* \*node)

Delete an SVG DOM subtree.

**Parametri**

- **node** -- pointer to an SVG DOM subtree

struct **lv\_svg\_point\_t**

## Public Members

float **x**

float **y**

struct **lv\_svg\_matrix\_t**

## Public Members

float **m[3][3]**

union **lv\_svg\_attr\_value\_t**

## Public Members

int32\_t **ival**

uint32\_t **uval**

```
float fval  
char *sval  
void *val  
  
struct lv_svg_attr_values_list_t
```

#### Public Members

```
uint32_t length  
  
uint8_t data[1]  
  
struct lv_svg_attr_path_value_t
```

#### Public Members

```
uint32_t cmd  
  
uint8_t data[1]  
  
struct lv_svg_attr_t
```

#### Public Members

```
lv_svg_attr_type_t id  
  
lv_svg_attr_value_type_t val_type  
  
lv_svg_attr_value_class_t class_type  
  
lv_svg_attr_value_t value
```

```
struct lv_svg_node_t
```

#### Public Members

```
lv_tree_node_t base  
  
char *xml_id  
  
lv_svg_tag_t type  
  
lv_array_t attrs  
  
struct lv_svg_render_obj *render_obj
```

**lv\_svg\_decoder.h****Functions**

**void lv\_svg\_decoder\_init(void)**  
 Register the SVG decoder functions in LVGL

**void lv\_svg\_decoder\_deinit(void)**

**lv\_svg\_parser.h****Enums****enum \_lv\_svg\_parser\_state\_t**

*Values:*

enumerator **LV\_SVG\_PARSER\_PROCESS**

enumerator **LV\_SVG\_PARSER\_IGNORE**

**Functions**

**void \_lv\_svg\_parser\_init(\_lv\_svg\_parser\_t \*parser)**

Initialize the SVG parser.

**Parametri**

**parser** -- pointer to a parser object

**void \_lv\_svg\_parser\_deinit(\_lv\_svg\_parser\_t \*parser)**

Deinitialize the SVG parser.

**Parametri**

**parser** -- pointer to a parser object

**bool \_lv\_svg\_parser\_token(\_lv\_svg\_parser\_t \*parser, const \_lv\_svg\_token\_t \*token)**

Parse an SVG document.

**Parametri**

- **parser** -- pointer to a parser object
- **token** -- pointer to a token object

**Ritorna**

true: the parsing is finished, false: the parsing is not finished yet.

**bool \_lv\_svg\_parser\_is\_finish(\_lv\_svg\_parser\_t \*parser)**

Check if the parsing is finished.

**Parametri**

**parser** -- pointer to a parser object

**Ritorna**

true: the parsing is finished, false: the parsing is not finished yet.

**void \_lv\_svg\_dump\_tree(\_lv\_svg\_node\_t \*root, int depth)**

Dump the SVG tree.

**Parametri**

- **root** -- pointer to the root of the SVG tree
- **depth** -- the depth of the current node in the tree

**struct \_lv\_svg\_parser\_t**

**Public Members**

```
uint16_t state  

char *ignore_name  

uint32_t ignore_len  

int32_t dpi  

lv_svg_node_t *doc_root  

lv_svg_node_t *cur_node
```

**lv\_svg\_render.h****Defines****LV\_SVG\_RENDER\_OBJ(n)****Typedef**

```
typedef struct _lv_svg_render_obj lv_svg_render_obj_t  

typedef struct _lv_svg_render_class lv_svg_render_class  

typedef struct _lv_svg_render_hal lv_svg_render_hal_t
```

**Functions****void lv\_svg\_render\_init(const lv\_svg\_render\_hal\_t \*hal)**

Initialize the SVG render.

**Parametri****hal** -- pointer to a structure with rendering functions**lv\_svg\_render\_obj\_t \*lv\_svg\_render\_create(const lv\_svg\_node\_t \*svg\_doc)**

Create a new SVG render from an SVG document.

**Parametri****svg\_doc** -- pointer to the SVG document**Ritorna**

pointer to the new SVG render object

**void lv\_svg\_render\_delete(lv\_svg\_render\_obj\_t \*render)**

Delete an SVG render object.

**Parametri****render** -- pointer to the SVG render object to delete**uint32\_t lv\_svg\_render\_get\_size(const lv\_svg\_render\_obj\_t \*render)**

Get size of render objects.

**Parametri****render** -- pointer to the SVG render object**Ritorna**

the bytes of SVG render objects

---

`lv_result_t lv_svg_render_get_viewport_size(const lv_svg_render_obj_t *render, float *width, float *height)`

Get viewport's width and height of the render object.

**Parametri**

- **render** -- pointer to the SVG render object
- **width** -- pointer to save the width of the viewport of the SVG render object
- **height** -- pointer to save the height of the viewport of the SVG render object

**Ritorna**

`lv_result_t, LV_RESULT_OK if success, LV_RESULT_INVALID if fail`

`void lv_draw_svg_render(lv_vector_dsc_t *dsc, const lv_svg_render_obj_t *render)`

Render an SVG object to a vector graphics.

**Parametri**

- **dsc** -- pointer to the vector graphics descriptor
- **render** -- pointer to the SVG render object to render

`void lv_draw_svg(lv_layer_t *layer, const lv_svg_node_t *svg_doc)`

Draw an SVG document to a layer.

**Parametri**

- **layer** -- pointer to the target layer
- **svg\_doc** -- pointer to the SVG document to draw

`struct _lv_svg_render_obj`

### Public Members

`struct _lv_svg_render_obj *next`

`lv_svg_tag_t tag`

`uint32_t flags`

`char *id`

`lv_vector_draw_dsc_t dsc`

`lv_matrix_t matrix`

`struct _lv_svg_render_obj *head`

`char *fill_ref`

`char *stroke_ref`

`struct _lv_svg_render_class *cls`

`struct _lv_svg_render_class`

**Public Members**

```
void (*set_paint_ref)(struct lv_svg_render_obj *obj, lv_vector_draw_dsc_t *dsc, const struct lv_svg_render_obj *target_obj, bool fill)

void (*init)(struct lv_svg_render_obj *obj, const lv_svg_node_t *node)

void (*render)(const struct lv_svg_render_obj *obj, lv_vector_dsc_t *dsc, const lv_matrix_t *matrix)

void (*set_attr)(struct lv_svg_render_obj *obj, lv_vector_draw_dsc_t *dsc, const lv_svg_attr_t *attr)

void (*get_bounds)(const struct lv_svg_render_obj *obj, lv_area_t *area)

void (*get_size)(const struct lv_svg_render_obj *obj, uint32_t *size)

void (*destroy)(struct lv_svg_render_obj *obj)
```

struct ***lv\_svg\_render\_hal*****Public Members**

```
void (*load_image)(const char *image_url, lv_draw_image_dsc_t *img_dsc)

const char *(*get_font_path)(const char *font_family)
```

**lv\_svg\_token.h****Defines****SVG\_TOKEN\_LEN(t)****Typedef**

typedef bool (\***svg\_token\_process**)(*lv\_svg\_token\_t* \*token, void \*user\_data)

**Enums****enum *lv\_svg\_token\_type\_t***

*Values:*

enumerator **LV\_SVG\_TOKEN\_BEGIN**

enumerator **LV\_SVG\_TOKEN\_END**

enumerator **LV\_SVG\_TOKEN\_CONTENT**

**Functions**

```
bool _lv_svg_tokenizer(const char *svg_data, uint32_t len, svg_token_process cb, void *user_data)
```

Parse SVG data and call a callback for each token.

**Parametri**

- **svg\_data** -- pointer to SVG data
- **len** -- length of the SVG data
- **cb** -- callback function to be called for each token
- **user\_data** -- custom data to be passed to the callback function

**Ritorna**

true: SVG data successfully parsed, false: error occurred

```
struct _lv_svg_token_attr_t
```

**Public Members**

```
const char *name_start
```

```
const char *name_end
```

```
const char *value_start
```

```
const char *value_end
```

```
struct _lv_svg_token_t
```

**Public Members**

```
const char *start
```

```
const char *end
```

```
lv_svg_token_type_t type
```

```
bool flat
```

```
lv_svg_token_attr_t *cur_attr
```

```
lv_array_t attrs
```

**14.17.15 tiny\_ttf**

**lv\_tiny\_ttf.h**

**Functions**

```
lv_font_t *lv_tiny_ttf_create_data(const void *data, size_t data_size, int32_t font_size)
```

Create a font from the specified data pointer with the specified line height.

**Parametri**

- **data** -- the data pointer
- **data\_size** -- the data size
- **font\_size** -- the font size in pixel

**Ritorna**

a font object

```
lv_font_t *lv_tiny_ttf_create_data_ex(const void *data, size_t data_size, int32_t font_size,
                                       lv_font_kerning_t kerning, size_t cache_size)
```

Create a font from the specified data pointer with the specified line height and the specified cache size.

**Parametri**

- **data** -- the data pointer
- **data\_size** -- the data size
- **font\_size** -- the font size in pixel
- **kerning** -- kerning value in pixel
- **cache\_size** -- the cache size in count

**Ritorna**

```
void lv_tiny_ttf_set_size(lv_font_t *font, int32_t font_size)
```

Set the size of the font to a new font\_size

 **Nota**

the font bitmap cache and glyph cache will be flushed.

**Parametri**

- **font** -- the font object
- **font\_size** -- the font size in pixel

```
void lv_tiny_ttf_destroy(lv_font_t *font)
```

Destroy a font previously created with lv\_tiny\_ttf\_create\_xxxx()

**Parametri**

**font** -- the font object

**Variables**

const lv\_font\_class\_t **lv\_tiny\_ttf\_font\_class**

struct **lv\_tiny\_ttf\_font\_src\_t**

**Public Members**

const char \***path**

Path to the font file

const void \***data**

Pointer to the font data

size\_t **data\_size**

Size of the font data

size\_t **cache\_size**

Size of the font cache

**14.17.16 tjpgd**

`lv_tjpngd.h`

## 14.18 misc

### 14.18.1 lv\_anim.h

Defines

`LV_ANIM_REPEAT_INFINITE`

`LV_ANIM_PLAYTIME_INFINITE`

`LV_ANIM_PAUSE_FOREVER`

`_PARA(a, x1, y1, x2, y2)`

`LV_ANIM_SET_EASE_IN_SINE(a)`

`LV_ANIM_SET_EASE_OUT_SINE(a)`

`LV_ANIM_SET_EASE_IN_OUT_SINE(a)`

`LV_ANIM_SET_EASE_IN_QUAD(a)`

`LV_ANIM_SET_EASE_OUT_QUAD(a)`

`LV_ANIM_SET_EASE_IN_OUT_QUAD(a)`

`LV_ANIM_SET_EASE_IN_CUBIC(a)`

`LV_ANIM_SET_EASE_OUT_CUBIC(a)`

`LV_ANIM_SET_EASE_IN_OUT_CUBIC(a)`

`LV_ANIM_SET_EASE_IN_QUART(a)`

`LV_ANIM_SET_EASE_OUT_QUART(a)`

`LV_ANIM_SET_EASE_IN_OUT_QUART(a)`

`LV_ANIM_SET_EASE_IN_QUINT(a)`

`LV_ANIM_SET_EASE_OUT_QUINT(a)`

`LV_ANIM_SET_EASE_IN_OUT_QUINT(a)`

`LV_ANIM_SET_EASE_IN_EXPO(a)`

`LV_ANIM_SET_EASE_OUT_EXPO(a)`

`LV_ANIM_SET_EASE_IN_OUT_EXPO(a)`

`LV_ANIM_SET_EASE_IN_CIRC(a)`

`LV_ANIM_SET_EASE_OUT_CIRC(a)`

`LV_ANIM_SET_EASE_IN_OUT_CIRC(a)`

`LV_ANIM_SET_EASE_IN_BACK(a)`

`LV_ANIM_SET_EASE_OUT_BACK(a)`

**LV\_ANIM\_SET\_EASE\_IN\_OUT\_BACK(a)**

**LV\_ANIM\_OFF**

Can be used to indicate if animations are enabled or disabled in a case

**LV\_ANIM\_ON**

**Typedef**

typedef bool **lv\_anim\_enable\_t**

typedef int32\_t (\***lv\_anim\_path\_cb\_t**)(const lv\_anim\_t\*)

Get the current value during an animation

typedef void (\***lv\_anim\_exec\_xcb\_t**)(void\*, int32\_t)

Generic prototype of "animator" functions. First parameter is the variable to animate. Second parameter is the value to set. Compatible with `lv_xxx_set_yyy(obj, value)` functions. The X in `_xcb_t` means it's not a fully generic prototype because it doesn't receive `lv_anim_t *` as its first argument

typedef void (\***lv\_anim\_custom\_exec\_cb\_t**)(lv\_anim\_t\*, int32\_t)

Same as `lv_anim_exec_xcb_t` but receives `lv_anim_t *` as the first parameter. It's more consistent but less convenient. Might be used by binding generator functions.

typedef void (\***lv\_anim\_completed\_cb\_t**)(lv\_anim\_t\*)

Callback to call when the animation is ready

typedef void (\***lv\_anim\_start\_cb\_t**)(lv\_anim\_t\*)

Callback to call when the animation really stars (considering `delay`)

typedef int32\_t (\***lv\_anim\_get\_value\_cb\_t**)(lv\_anim\_t\*)

Callback used when the animation values are relative to get the current value

typedef void (\***lv\_anim\_deleted\_cb\_t**)(lv\_anim\_t\*)

Callback used when the animation is deleted

## Functions

void **lv\_anim\_init**(lv\_anim\_t \*a)

Initialize an animation variable. E.g.: `lv_anim_t a; lv_anim_init(&a); lv_anim_set....(&a); lv_anim_start(&a);`

### Parametri

- **a** -- pointer to an `lv_anim_t` variable to initialize

void **lv\_anim\_set\_var**(lv\_anim\_t \*a, void \*var)

Set a variable to animate

### Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable
- **var** -- pointer to a variable to animate

void **lv\_anim\_set\_exec\_cb**(lv\_anim\_t \*a, [lv\\_anim\\_exec\\_xcb\\_t](#) exec\_cb)

Set a function to animate `var`

### Parametri

- **a** -- pointer to an initialized `lv_anim_t` variable

- **exec\_cb** -- a function to execute during animation LVGL's built-in functions can be used. E.g. lv\_obj\_set\_x

**void lv\_anim\_set\_duration(lv\_anim\_t \*a, uint32\_t duration)**

Set the duration of an animation

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **duration** -- duration of the animation in milliseconds

**void lv\_anim\_set\_delay(lv\_anim\_t \*a, uint32\_t delay)**

Set a delay before starting the animation

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay before the animation in milliseconds

**void lv\_anim\_resume(lv\_anim\_t \*a)**

Resumes a paused animation

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable

**void lv\_anim\_pause(lv\_anim\_t \*a)**

Pauses the animation

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable

**void lv\_anim\_pause\_for(lv\_anim\_t \*a, uint32\_t ms)**

Pauses the animation for ms milliseconds

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **ms** -- the pause time in milliseconds

**bool lv\_anim\_is\_paused(lv\_anim\_t \*a)**

Check if the animation is paused

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable

**Ritorna**

true if the animation is paused else false

**void lv\_anim\_set\_values(lv\_anim\_t \*a, int32\_t start, int32\_t end)**

Set the start and end values of an animation

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **start** -- the start value
- **end** -- the end value

**void lv\_anim\_set\_custom\_exec\_cb(lv\_anim\_t \*a, lv\_anim\_custom\_exec\_cb\_t exec\_cb)**

Similar to `lv_anim_set_exec_cb` but `lv_anim_custom_exec_cb_t` receives `lv_anim_t *` as its first parameter instead of `void *`. This function might be used when LVGL is bound to other languages because it's more consistent to have `lv_anim_t *` as first parameter.

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **exec\_cb** -- a function to execute.

**void lv\_anim\_set\_path\_cb(lv\_anim\_t \*a, lv\_anim\_path\_cb\_t path\_cb)**

Set the path (curve) of the animation.

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **path\_cb** -- a function to set the current value of the animation.

**void lv\_anim\_set\_start\_cb(lv\_anim\_t \*a, lv\_anim\_start\_cb\_t start\_cb)**

Set a function call when the animation really starts (considering `delay`)

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **start\_cb** -- a function call when the animation starts

`void lv_anim_set_get_value_cb(lv_anim_t *a, lv_anim_get_value_cb_t get_value_cb)`

Set a function to use the current value of the variable and make start and end value relative to the returned current value.

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **get\_value\_cb** -- a function call when the animation starts

`void lv_anim_set_completed_cb(lv_anim_t *a, lv_anim_completed_cb_t completed_cb)`

Set a function call when the animation is completed

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **completed\_cb** -- a function call when the animation is fully completed

`void lv_anim_set_deleted_cb(lv_anim_t *a, lv_anim_deleted_cb_t deleted_cb)`

Set a function call when the animation is deleted.

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **deleted\_cb** -- a function call when the animation is deleted

`void lv_anim_set_reverse_duration(lv_anim_t *a, uint32_t duration)`

Make the animation to play back to when the forward direction is ready

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **duration** -- duration of playback animation in milliseconds. 0: disable playback

`void lv_anim_set_reverse_time(lv_anim_t *a, uint32_t duration)`

Legacy `lv_anim_set_reverse_time` API will be removed soon, use `lv_anim_set_reverse_duration` instead.

`void lv_anim_set_reverse_delay(lv_anim_t *a, uint32_t delay)`

Make the animation to play back to when the forward direction is ready

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay in milliseconds before starting the playback animation.

`void lv_anim_set_repeat_count(lv_anim_t *a, uint32_t cnt)`

Make the animation repeat itself.

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **cnt** -- repeat count or `LV_ANIM_REPEAT_INFINITE` for infinite repetition. 0: to disable repetition.

`void lv_anim_set_repeat_delay(lv_anim_t *a, uint32_t delay)`

Set a delay before repeating the animation.

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay in milliseconds before repeating the animation.

`void lv_anim_set_early_apply(lv_anim_t *a, bool en)`

Set a whether the animation's should be applied immediately or only when the delay expired.

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **en** -- true: apply the start value immediately in `lv_anim_start`; false: apply the start value only when `delay` ms is elapsed and the animations really starts

`void lv_anim_set_user_data(lv_anim_t *a, void *user_data)`

Set the custom user data field of the animation.

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **user\_data** -- pointer to the new user\_data.

**void `lv_anim_set_bezier3_param`(`lv_anim_t` \*a, `int16_t` x1, `int16_t` y1, `int16_t` x2, `int16_t` y2)**

Set parameter for cubic bezier path

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable
- **x1** -- first control point X
- **y1** -- first control point Y
- **x2** -- second control point X
- **y2** -- second control point Y

**lv\_anim\_t \*`lv_anim_start`(const `lv_anim_t` \*a)**

Create an animation

**Parametri**

- **a** -- an initialized 'anim\_t' variable. Not required after call.

**Ritorna**

- pointer to the created animation (different from the **a** parameter)

**uint32\_t `lv_anim_get_delay`(const `lv_anim_t` \*a)**

Get a delay before starting the animation

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable

**Ritorna**

- delay before the animation in milliseconds

**uint32\_t `lv_anim_get_playtime`(const `lv_anim_t` \*a)**

Get the time used to play the animation.

**Parametri**

- **a** -- pointer to an animation.

**Ritorna**

- the play time in milliseconds.

**uint32\_t `lv_anim_get_time`(const `lv_anim_t` \*a)**

Get the duration of an animation

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable

**Ritorna**

- the duration of the animation in milliseconds

**uint32\_t `lv_anim_get_repeat_count`(const `lv_anim_t` \*a)**

Get the repeat count of the animation.

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable

**Ritorna**

- the repeat count or `LV_ANIM_REPEAT_INFINITE` for infinite repetition. 0: disabled repetition.

**void \*`lv_anim_get_user_data`(const `lv_anim_t` \*a)**

Get the user\_data field of the animation

**Parametri**

- **a** -- pointer to an initialized `lv_anim_t` variable

**Ritorna**

- the pointer to the custom user\_data of the animation

**bool `lv_anim_delete`(void \*var, `lv_anim_exec_xcb_t` exec\_cb)**

Delete animation(s) of a variable with a given animator function

**Parametri**

- **var** -- pointer to variable

- **exec\_cb** -- a function pointer which is animating 'var', or NULL to ignore it and delete all the animations of 'var'

**Ritorna**

true: at least 1 animation is deleted, false: no animation is deleted

**void lv\_anim\_delete\_all(void)**

Delete all the animations

**lv\_anim\_t \*lv\_anim\_get(void \*var, lv\_anim\_exec\_xcb\_t exec\_cb)**

Get the animation of a variable and its **exec\_cb**.

**Parametri**

- **var** -- pointer to variable
- **exec\_cb** -- a function pointer which is animating 'var', or NULL to return first matching 'var'

**Ritorna**

pointer to the animation.

**lv\_timer\_t \*lv\_anim\_get\_timer(void)**

Get global animation refresher timer.

**Ritorna**

pointer to the animation refresher timer.

**bool lv\_anim\_custom\_delete(lv\_anim\_t \*a, lv\_anim\_custom\_exec\_cb\_t exec\_cb)**

Delete an animation by getting the animated variable from **a**. Only animations with **exec\_cb** will be deleted. This function exists because it's logical that all anim. functions receives an **lv\_anim\_t** as their first parameter. It's not practical in C but might make the API more consequent and makes easier to generate bindings.

**Parametri**

- **a** -- pointer to an animation.
- **exec\_cb** -- a function pointer which is animating 'var', or NULL to ignore it and delete all the animations of 'var'

**Ritorna**

true: at least 1 animation is deleted, false: no animation is deleted

**lv\_anim\_t \*lv\_anim\_custom\_get(lv\_anim\_t \*a, lv\_anim\_custom\_exec\_cb\_t exec\_cb)**

Get the animation of a variable and its **exec\_cb**. This function exists because it's logical that all anim. functions receives an **lv\_anim\_t** as their first parameter. It's not practical in C but might make the API more consequent and makes easier to generate bindings.

**Parametri**

- **a** -- pointer to an animation.
- **exec\_cb** -- a function pointer which is animating 'var', or NULL to return first matching 'var'

**Ritorna**

pointer to the animation.

**uint16\_t lv\_anim\_count\_running(void)**

Get the number of currently running animations

**Ritorna**

the number of running animations

**uint32\_t lv\_anim\_speed(uint32\_t speed)**

Store the speed as a special value which can be used as time in animations. It will be converted to time internally based on the start and end values. The return value can be used as a constant with multiple animations and let LVGL convert the speed to time based on the actual values. LIMITATION: the max time stored this way can be 10,000 ms.

 **Nota**

internally speed is stored as 10 unit/sec

**Parametri**

- **speed** -- the speed of the animation in with unit / sec resolution in 0..10k range

**Ritorna**

a special value which can be used as an animation time

```
uint32_t lv_anim_speed_clamped(uint32_t speed, uint32_t min_time, uint32_t max_time)
```

Store the speed as a special value which can be used as time in animations. It will be converted to time internally based on the start and end values. The return value can be used as a constant with multiple animations and let LVGL convert the speed to time based on the actual values.

 **Nota**

internally speed is stored as 10 unit/sec

 **Nota**

internally min/max\_time are stored with 10 ms unit

**Parametri**

- **speed** -- the speed of the animation in as unit / sec resolution in 0..10k range
- **min\_time** -- the minimum time in 0..10k range
- **max\_time** -- the maximum time in 0..10k range

**Ritorna**

a special value in where all three values are stored and can be used as an animation time

```
uint32_t lv_anim_resolve_speed(uint32_t speed, int32_t start, int32_t end)
```

Resolve the speed (created with `lv_anim_speed` or `lv_anim_speed_clamped`) to time based on start and end values.

**Parametri**

- **speed** -- return values of `lv_anim_speed` or `lv_anim_speed_clamped`
- **start** -- the start value of the animation
- **end** -- the end value of the animation

**Ritorna**

the time required to get from `start` to `end` with the given `speed` setting

```
uint32_t lv_anim_speed_to_time(uint32_t speed, int32_t start, int32_t end)
```

Calculate the time of an animation based on its speed, start and end values. It simpler than `lv_anim_speed` or `lv_anim_speed_clamped` as it converts speed, start, and end to a time immediately. As it's simpler there is no limit on the maximum time.

**Parametri**

- **speed** -- the speed of the animation
- **start** -- the start value
- **end** -- the end value

**Ritorna**

the time of the animation in milliseconds

```
void lv_anim_refr_now(void)
```

Manually refresh the state of the animations. Useful to make the animations running in a blocking process where `lv_timer_handler` can't run for a while. Shouldn't be used directly because it is called in `lv_refr_now()`.

```
int32_t lv_anim_path_linear(const lv_anim_t *a)
```

Calculate the current value of an animation applying linear characteristic

**Parametri**

- **a** -- pointer to an animation

**Ritorna**

the current value to set

`int32_t lv_anim_path_ease_in(const lv_anim_t *a)`

Calculate the current value of an animation slowing down the start phase

**Parametri**

**a** -- pointer to an animation

**Ritorna**

the current value to set

`int32_t lv_anim_path_ease_out(const lv_anim_t *a)`

Calculate the current value of an animation slowing down the end phase

**Parametri**

**a** -- pointer to an animation

**Ritorna**

the current value to set

`int32_t lv_anim_path_ease_in_out(const lv_anim_t *a)`

Calculate the current value of an animation applying an "S" characteristic (cosine)

**Parametri**

**a** -- pointer to an animation

**Ritorna**

the current value to set

`int32_t lv_anim_path_overshoot(const lv_anim_t *a)`

Calculate the current value of an animation with overshoot at the end

**Parametri**

**a** -- pointer to an animation

**Ritorna**

the current value to set

`int32_t lv_anim_path_bounce(const lv_anim_t *a)`

Calculate the current value of an animation with 3 bounces

**Parametri**

**a** -- pointer to an animation

**Ritorna**

the current value to set

`int32_t lv_anim_path_step(const lv_anim_t *a)`

Calculate the current value of an animation applying step characteristic. (Set end value on the end of the animation)

**Parametri**

**a** -- pointer to an animation

**Ritorna**

the current value to set

`int32_t lv_anim_path_custom_bezier3(const lv_anim_t *a)`

A custom cubic bezier animation path, need to specify cubic-parameters in a->parameter.bezier3

**Parametri**

**a** -- pointer to an animation

**Ritorna**

the current value to set

`struct lv_anim_bezier3_para_t`

#include <lv\_anim.h> Parameter used when path is custom\_bezier

### Public Members

`int16_t x1`

```
int16_t y1  
  
int16_t x2  
  
int16_t y2  
  
struct _lv_anim_t  
#include <lv_anim.h> Describes an animation
```

### Public Members

**void \*var**

Variable (Widget or other user-provided object) to animate

*lv\_anim\_exec\_xcb\_t exec\_cb*

Function to execute to animate

*lv\_anim\_custom\_exec\_cb\_t custom\_exec\_cb*

Function to execute to animate, same purpose as exec\_cb but different parameters

*lv\_anim\_start\_cb\_t start\_cb*

Call it when animation is starts (considering delay)

*lv\_anim\_completed\_cb\_t completed\_cb*

Call it when animation is fully completed

*lv\_anim\_deleted\_cb\_t deleted\_cb*

Call it when animation is deleted

*lv\_anim\_get\_value\_cb\_t get\_value\_cb*

Get current value in relative mode

**void \*user\_data**

Custom user data

*lv\_anim\_path\_cb\_t path\_cb*

Provides path (curve) of animation

**int32\_t start\_value**

Start value

**int32\_t current\_value**

Current value

**int32\_t end\_value**

End value

**int32\_t duration**

Animation duration in ms

**int32\_t act\_time**

Ms elapsed since animation started. Set to negative to make delay.

**uint32\_t reverse\_delay**

Wait (in ms) after forward play ends and before reverse play begins.

**uint32\_t reverse\_duration**

Reverse animation duration in ms

**uint32\_t repeat\_delay**

Wait before repeating

**uint32\_t repeat\_cnt**

Repeat count for animation

**union *lv\_anim\_t*::*lv\_anim\_path\_para\_t* parameter****uint32\_t last\_timer\_run****uint32\_t pause\_time**

The time when the animation was paused

**uint32\_t pause\_duration**

The amount of the time the animation must stay paused for

**uint8\_t is\_paused**

Indicates that the animation is paused

**uint8\_t reverse\_play\_in\_progress**

Reverse play is in progress

**uint8\_t run\_round**

When not equal to global.anim\_state.anim\_run\_round (which toggles each time animation timer executes), indicates this animation needs to be updated.

**uint8\_t start\_cb\_called**

Indicates that `start_cb` was already called

**uint8\_t early\_apply**

1: Apply start value immediately even if there is a `delay`

**union *lv\_anim\_path\_para\_t*****Public Members*****lv\_anim\_bezier3\_para\_t* bezier3**

Parameter used when path is `custom_bezier`

## 14.18.2 lv\_anim\_private.h

### Functions

```
void lv_anim_core_init(void)
    Init the animation module

void lv_anim_core_deinit(void)
    Deinit the animation module

void lv_anim_enable_vsync_mode(bool enable)

struct lv_anim_state_t
```

### Public Members

```
bool anim_list_changed

bool anim_run_round

bool anim_vsync_registered

lv_timer_t *timer

lv_ll_t anim_ll
```

## 14.18.3 lv\_anim\_timeline.h

### Defines

**LV\_ANIM\_TIMELINE\_PROGRESS\_MAX**

### Typedef

```
typedef struct _lv_anim_timeline_dsc_t lv_anim_timeline_dsc_t
```

### Functions

`lv_anim_timeline_t *lv_anim_timeline_create(void)`

Create an animation timeline.

#### Ritorna

pointer to the animation timeline.

`void lv_anim_timeline_delete(lv_anim_timeline_t *at)`

Delete animation timeline.

#### Parametri

**at** -- pointer to the animation timeline.

`void lv_anim_timeline_add(lv_anim_timeline_t *at, uint32_t start_time, const lv_anim_t *a)`

Add animation to the animation timeline.

#### Parametri

- **at** -- pointer to the animation timeline.
- **start\_time** -- the time the animation started on the timeline, note that start\_time will override the value of delay.
- **a** -- pointer to an animation.

`uint32_t lv_anim_timeline_start(lv_anim_timeline_t *at)`

Start the animation timeline.

**Parametri**

- **at** -- pointer to the animation timeline.

**Ritorna**

total time spent in animation timeline.

`void lv_anim_timeline_pause(lv_anim_timeline_t *at)`

Pause the animation timeline.

**Parametri**

- **at** -- pointer to the animation timeline.

`void lv_anim_timeline_set_reverse(lv_anim_timeline_t *at, bool reverse)`

Set the playback direction of the animation timeline.

**Parametri**

- **at** -- pointer to the animation timeline.
- **reverse** -- whether to play in reverse.

`void lv_anim_timeline_set_delay(lv_anim_timeline_t *at, uint32_t delay)`

Set the time to wait before starting the the animation. Applies only when playing from the very start, or reverse from the very end.

**Parametri**

- **at** -- pointer to an animation timeline
- **delay** -- the delay time in milliseconds

`void lv_anim_timeline_set_repeat_count(lv_anim_timeline_t *at, uint32_t cnt)`

Make the animation timeline repeat itself.

**Parametri**

- **at** -- pointer to the animation timeline.
- **cnt** -- repeat count or `LV_ANIM_REPEAT_INFINITE` for infinite repetition. 0: to disable repetition.

`void lv_anim_timeline_set_repeat_delay(lv_anim_timeline_t *at, uint32_t delay)`

Set a delay before repeating the animation timeline.

**Parametri**

- **at** -- pointer to the animation timeline.
- **delay** -- delay in milliseconds before repeating the animation timeline.

`void lv_anim_timeline_set_progress(lv_anim_timeline_t *at, uint16_t progress)`

Set the progress of the animation timeline.

**Parametri**

- **at** -- pointer to the animation timeline.
- **progress** -- set value 0~65535 to map 0~100% animation progress.

`void lv_anim_timeline_set_user_data(lv_anim_timeline_t *at, void *user_data)`

Set the user\_data of a an animation timeline

**Parametri**

- **at** -- pointer to the animation timeline.
- **user\_data** -- pointer to any data. Only the pointer will be saved.

`void lv_anim_timeline_set_base_obj(lv_anim_timeline_t *at, lv_obj_t *base_obj)`

Set base object. If set, it's assumed that the `var` of animations is a widget name (path). The widget pointer will be retrieved by finding them by name on this widget.

**Parametri**

- **at** -- pointer to the animation timeline.
- **base\_obj** -- pointer to a widget

`uint32_t lv_anim_timeline_get_playtime(lv_anim_timeline_t *at)`

Get the time used to play the animation timeline.

**Parametri**

- **at** -- pointer to the animation timeline.

**Ritorna**

total time spent in animation timeline.

**bool lv\_anim\_timeline\_get\_reverse(lv\_anim\_timeline\_t \*at)**

Get whether the animation timeline is played in reverse.

**Parametri**

**at** -- pointer to the animation timeline.

**Ritorna**

return true if it is reverse playback.

**uint32\_t lv\_anim\_timeline\_get\_delay(lv\_anim\_timeline\_t \*at)**

Get the wait time when playing from the very start, or reverse from the very end.

**Parametri**

**at** -- pointer to an animation timeline

**Ritorna**

the remaining time in milliseconds

**uint16\_t lv\_anim\_timeline\_get\_progress(lv\_anim\_timeline\_t \*at)**

Get the progress of the animation timeline.

**Parametri**

**at** -- pointer to the animation timeline.

**Ritorna**

return value 0~65535 to map 0~100% animation progress.

**uint32\_t lv\_anim\_timeline\_get\_repeat\_count(lv\_anim\_timeline\_t \*at)**

Get repeat count of the animation timeline.

**Parametri**

**at** -- pointer to the animation timeline.

**uint32\_t lv\_anim\_timeline\_get\_repeat\_delay(lv\_anim\_timeline\_t \*at)**

Get repeat delay of the animation timeline.

**Parametri**

**at** -- pointer to the animation timeline.

**void \*lv\_anim\_timeline\_get\_user\_data(lv\_anim\_timeline\_t \*at)**

Get the user\_data of a an animation timeline

**Parametri**

**at** -- pointer to the animation timeline.

**lv\_obj\_t \*lv\_anim\_timeline\_get\_base\_obj(lv\_anim\_timeline\_t \*at)**

Get base object. If set, it's assumed that the var of animations is a widget name (path). The widget pointer will be retrieved by finding them by name on this widget.

**Parametri**

**at** -- pointer to the animation timeline.

**Ritorna**

pointer to the base widget

**struct \_lv\_anim\_timeline\_dsc\_t**

**Public Members**

**lv\_anim\_t anim**

**uint32\_t start\_time**

**uint8\_t is\_started**

**uint8\_t is\_completed**

## 14.18.4 lv\_anim\_timeline\_private.h

`struct _lv_anim_timeline_t`

### Public Members

`struct _lv_anim_timeline_dsc_t *anim_dsc`

Dynamically allocated anim dsc array

`uint32_t anim_dsc_cnt`

The length of anim dsc array

`uint32_t act_time`

Current time of the animation

`bool reverse`

Reverse playback

`uint32_t delay`

Delay before starting the animation from any ends

`uint32_t repeat_count`

Repeat count

`uint32_t repeat_delay`

Wait before repeat

`lv_obj_t *base_obj`

If set, it's assumed that the `var` of animations is a widget name (path). The widget pointer will be retrieved by finding them by name on this widget.

`void *user_data`

For any custom data

## 14.18.5 lv\_area.h

### Defines

`LV_COORD_TYPE_SHIFT`

`LV_COORD_TYPE_MASK`

`LV_COORD_TYPE(x)`

`LV_COORD_PLAIN(x)`

`LV_COORD_TYPE_PX`

`LV_COORD_TYPE_SPEC`

**LV\_COORD\_TYPE\_PX\_NEG**

**LV\_COORD\_IS\_PX(x)**

**LV\_COORD\_IS\_SPEC(x)**

**LV\_COORD\_SET\_SPEC(x)**

**LV\_COORD\_MAX**

Max coordinate value

**LV\_COORD\_MIN**

**LV\_SIZE\_CONTENT**

**LV\_PCT\_STORED\_MAX**

**LV\_PCT\_POS\_MAX**

**LV\_PCT(x)**

**LV\_COORD\_IS\_PCT(x)**

**LV\_COORD\_GET\_PCT(x)**

## Enums

enum **lv\_align\_t**

Alignments

*Values:*

enumerator **LV\_ALIGN\_DEFAULT**

enumerator **LV\_ALIGN\_TOP\_LEFT**

enumerator **LV\_ALIGN\_TOP\_MID**

enumerator **LV\_ALIGN\_TOP\_RIGHT**

enumerator **LV\_ALIGN\_BOTTOM\_LEFT**

enumerator **LV\_ALIGN\_BOTTOM\_MID**

enumerator **LV\_ALIGN\_BOTTOM\_RIGHT**

enumerator **LV\_ALIGN\_LEFT\_MID**

enumerator **LV\_ALIGN\_RIGHT\_MID**

enumerator **LV\_ALIGN\_CENTER**

enumerator **LV\_ALIGN\_OUT\_TOP\_LEFT**

enumerator **LV\_ALIGN\_OUT\_TOP\_MID**

enumerator **LV\_ALIGN\_OUT\_TOP\_RIGHT**

enumerator **LV\_ALIGN\_OUT\_BOTTOM\_LEFT**

enumerator **LV\_ALIGN\_OUT\_BOTTOM\_MID**

enumerator **LV\_ALIGN\_OUT\_BOTTOM\_RIGHT**

enumerator **LV\_ALIGN\_OUT\_LEFT\_TOP**

enumerator **LV\_ALIGN\_OUT\_LEFT\_MID**

enumerator **LV\_ALIGN\_OUT\_LEFT\_BOTTOM**

enumerator **LV\_ALIGN\_OUT\_RIGHT\_TOP**

enumerator **LV\_ALIGN\_OUT\_RIGHT\_MID**

enumerator **LV\_ALIGN\_OUT\_RIGHT\_BOTTOM**

enum **lv\_dir\_t**

*Values:*

enumerator **LV\_DIR\_NONE**

enumerator **LV\_DIR\_LEFT**

enumerator **LV\_DIR\_RIGHT**

enumerator **LV\_DIR\_TOP**

enumerator **LV\_DIR\_BOTTOM**

enumerator **LV\_DIR\_HOR**

enumerator **LV\_DIR\_VER**

enumerator **LV\_DIR\_ALL**

## Functions

`void lv_area_set(lv_area_t *area_p, int32_t x1, int32_t y1, int32_t x2, int32_t y2)`

Initialize an area

**Parametri**

- **area\_p** -- pointer to an area
- **x1** -- left coordinate of the area
- **y1** -- top coordinate of the area
- **x2** -- right coordinate of the area
- **y2** -- bottom coordinate of the area

`static inline void lv_area_copy(lv_area_t *dest, const lv_area_t *src)`

Copy an area

**Parametri**

- **dest** -- pointer to the destination area
- **src** -- pointer to the source area

`int32_t lv_area_get_width(const lv_area_t *area_p)`

Get the width of an area

**Parametri**

**area\_p** -- pointer to an area

**Ritorna**

the width of the area (if  $x1 == x2 \rightarrow \text{width} = 1$ )

`int32_t lv_area_get_height(const lv_area_t *area_p)`

Get the height of an area

**Parametri**

**area\_p** -- pointer to an area

**Ritorna**

the height of the area (if  $y1 == y2 \rightarrow \text{height} = 1$ )

`void lv_area_set_width(lv_area_t *area_p, int32_t w)`

Set the width of an area

**Parametri**

- **area\_p** -- pointer to an area
- **w** -- the new width of the area ( $w == 1$  makes  $x1 == x2$ )

`void lv_area_set_height(lv_area_t *area_p, int32_t h)`

Set the height of an area

**Parametri**

- **area\_p** -- pointer to an area
- **h** -- the new height of the area ( $h == 1$  makes  $y1 == y2$ )

`uint32_t lv_area_get_size(const lv_area_t *area_p)`

Return with area of an area ( $x * y$ )

**Parametri**

**area\_p** -- pointer to an area

**Ritorna**

size of area

`void lv_area_increase(lv_area_t *area, int32_t w_extra, int32_t h_extra)`

`void lv_area_move(lv_area_t *area, int32_t x_ofs, int32_t y_ofs)`

`void lv_area_align(const lv_area_t *base, lv_area_t *to_align, lv_align_t align, int32_t ofs_x, int32_t ofs_y)`

Align an area to another

**Parametri**

- **base** -- an area where the other will be aligned
- **to\_align** -- the area to align
- **align** -- LV\_ALIGN\_...

- **ofs\_x** -- X offset
- **ofs\_y** -- Y offset

```
void lv_point_transform(lv_point_t *point, int32_t angle, int32_t scale_x, int32_t scale_y, const lv_point_t *pivot, bool zoom_first)
```

Transform a point

**Parametri**

- **point** -- pointer to a point
- **angle** -- angle with 0.1 resolutions (123 means 12.3°)
- **scale\_x** -- horizontal zoom, 256 means 100%
- **scale\_y** -- vertical zoom, 256 means 100%
- **pivot** -- pointer to the pivot point of the transformation
- **zoom\_first** -- true: zoom first and rotate after that; else: opposite order

```
void lv_point_array_transform(lv_point_t *points, size_t count, int32_t angle, int32_t scale_x, int32_t scale_y, const lv_point_t *pivot, bool zoom_first)
```

Transform an array of points

**Parametri**

- **points** -- pointer to an array of points
- **count** -- number of points in the array
- **angle** -- angle with 0.1 resolutions (123 means 12.3°)
- **scale\_x** -- horizontal zoom, 256 means 100%
- **scale\_y** -- vertical zoom, 256 means 100%
- **pivot** -- pointer to the pivot point of the transformation
- **zoom\_first** -- true: zoom first and rotate after that; else: opposite order

```
lv_point_t lv_point_from_precise(const lv_point_precise_t *p)
```

```
lv_point_precise_t lv_point_to_precise(const lv_point_t *p)
```

```
void lv_point_set(lv_point_t *p, int32_t x, int32_t y)
```

```
void lv_point_precise_set(lv_point_precise_t *p, lv_value_precise_t x, lv_value_precise_t y)
```

```
void lv_point_swap(lv_point_t *p1, lv_point_t *p2)
```

```
void lv_point_precise_swap(lv_point_precise_t *p1, lv_point_precise_t *p2)
```

```
int32_t lv_pct(int32_t x)
```

Convert a percentage value to `int32_t`. Percentage values are stored in special range

**Parametri**

**x** -- the percentage (0..1000)

**Ritorna**

a coordinate that stores the percentage

```
int32_t lv_pct_to_px(int32_t v, int32_t base)
```

```
struct lv_point_t
```

#include <lv\_area.h> Represents a point on the screen.

### Public Members

int32\_t x

int32\_t y

```
struct lv_point_precise_t
```

**Public Members**

`lv_value_precise_t x`

`lv_value_precise_t y`

struct **lv\_area\_t**

#include <lv\_area.h> Represents an area of the screen.

**Public Members**

`int32_t x1`

`int32_t y1`

`int32_t x2`

`int32_t y2`

**14.18.6 lv\_area\_private.h****Functions**

void **lv\_area\_set\_pos**(`lv_area_t` \*area\_p, `int32_t` x, `int32_t` y)

Set the position of an area (width and height will be kept)

**Parametri**

- **area\_p** -- pointer to an area
- **x** -- the new x coordinate of the area
- **y** -- the new y coordinate of the area

bool **lv\_area\_intersect**(`lv_area_t` \*res\_p, const `lv_area_t` \*a1\_p, const `lv_area_t` \*a2\_p)

Get the common parts of two areas

**Parametri**

- **res\_p** -- pointer to an area, the result will be stored here
- **a1\_p** -- pointer to the first area
- **a2\_p** -- pointer to the second area

**Ritorna**

false: the two area has NO common parts, res\_p is invalid

`int8_t lv_area_diff`(`lv_area_t` res\_p[], const `lv_area_t` \*a1\_p, const `lv_area_t` \*a2\_p)

Get resulting sub areas after removing the common parts of two areas from the first area

**Parametri**

- **res\_p** -- pointer to an array of areas with a count of 4, the resulting areas will be stored here
- **a1\_p** -- pointer to the first area
- **a2\_p** -- pointer to the second area

**Ritorna**

number of results (max 4) or -1 if no intersect

void **lv\_area\_join**(`lv_area_t` \*a\_res\_p, const `lv_area_t` \*a1\_p, const `lv_area_t` \*a2\_p)

Join two areas into a third which involves the other two

**Parametri**

- **a\_res\_p** -- pointer to an area, the result will be stored here
- **a1\_p** -- pointer to the first area
- **a2\_p** -- pointer to the second area

**bool lv\_area\_is\_point\_on(const lv\_area\_t \*a\_p, const lv\_point\_t \*p\_p, int32\_t radius)**

Check if a point is on an area

**Parametri**

- **a\_p** -- pointer to an area
- **p\_p** -- pointer to a point
- **radius** -- radius of area (e.g. for rounded rectangle)

**Ritorna**

false:the point is out of the area

**bool lv\_area\_is\_on(const lv\_area\_t \*a1\_p, const lv\_area\_t \*a2\_p)**

Check if two area has common parts

**Parametri**

- **a1\_p** -- pointer to an area.
- **a2\_p** -- pointer to another area

**Ritorna**

false: a1\_p and a2\_p has no common parts

**bool lv\_area\_is\_in(const lv\_area\_t \*ain\_p, const lv\_area\_t \*aholder\_p, int32\_t radius)**

Check if an area is fully on another

**Parametri**

- **ain\_p** -- pointer to an area which could be in 'aholder\_p'
- **aholder\_p** -- pointer to an area which could involve 'ain\_p'
- **radius** -- radius of aholder\_p (e.g. for rounded rectangle)

**Ritorna**

true: ain\_p is fully inside aholder\_p

**bool lv\_area\_is\_out(const lv\_area\_t \*aout\_p, const lv\_area\_t \*aholder\_p, int32\_t radius)**

Check if an area is fully out of another

**Parametri**

- **aout\_p** -- pointer to an area which could be in 'aholder\_p'
- **aholder\_p** -- pointer to an area which could involve 'ain\_p'
- **radius** -- radius of aholder\_p (e.g. for rounded rectangle)

**Ritorna**

true: aout\_p is fully outside aholder\_p

**bool lv\_area\_is\_equal(const lv\_area\_t \*a, const lv\_area\_t \*b)**

Check if 2 area is the same

**Parametri**

- **a** -- pointer to an area
- **b** -- pointer to another area

## 14.18.7 lv\_array.h

Array. The elements are dynamically allocated by the 'lv\_mem' module.

### Defines

**LV\_ARRAY\_DEFAULT\_CAPACITY**

**LV\_ARRAY\_DEFAULT\_SHRINK\_RATIO**

### Functions

**void lv\_array\_init(lv\_array\_t \*array, uint32\_t capacity, uint32\_t element\_size)**

Init an array.

**Parametri**

- **array** -- pointer to an **lv\_array\_t** variable to initialize
- **capacity** -- the initial capacity of the array

- **element\_size** -- the size of an element in bytes

**void lv\_array\_init\_from\_buf(lv\_array\_t \*array, void \*buf, uint32\_t capacity, uint32\_t element\_size)**

Init an array from a buffer.

#### **Nota**

The buffer must be large enough to store **capacity** elements. The array will not release the buffer and reallocate it. The user must ensure that the buffer is valid during the lifetime of the array. And release the buffer when the array is no longer needed.

#### **Parametri**

- **array** -- pointer to an **lv\_array\_t** variable to initialize
- **buf** -- pointer to a buffer to use as the array's data
- **capacity** -- the initial capacity of the array
- **element\_size** -- the size of an element in bytes

**bool lv\_array\_resize(lv\_array\_t \*array, uint32\_t new\_capacity)**

Resize the array to the given capacity.

#### **Nota**

if the new capacity is smaller than the current size, the array will be truncated.

#### **Parametri**

- **array** -- pointer to an **lv\_array\_t** variable
- **new\_capacity** -- the new capacity of the array

**void lv\_array\_deinit(lv\_array\_t \*array)**

Deinit the array, and free the allocated memory

#### **Parametri**

**array** -- pointer to an **lv\_array\_t** variable to deinitialize

**static inline uint32\_t lv\_array\_size(const lv\_array\_t \*array)**

Return how many elements are stored in the array.

#### **Parametri**

**array** -- pointer to an **lv\_array\_t** variable

#### **Ritorna**

the number of elements stored in the array

**static inline uint32\_t lv\_array\_capacity(const lv\_array\_t \*array)**

Return the capacity of the array, i.e. how many elements can be stored.

#### **Parametri**

**array** -- pointer to an **lv\_array\_t** variable

#### **Ritorna**

the capacity of the array

**static inline bool lv\_array\_is\_empty(const lv\_array\_t \*array)**

Return if the array is empty

#### **Parametri**

**array** -- pointer to an **lv\_array\_t** variable

#### **Ritorna**

true: array is empty; false: array is not empty

**static inline bool lv\_array\_is\_full(const lv\_array\_t \*array)**

Return if the array is full

#### **Parametri**

**array** -- pointer to an **lv\_array\_t** variable

**Ritorna**

true: array is full; false: array is not full

**void lv\_array\_copy(lv\_array\_t \*target, const lv\_array\_t \*source)**

Copy an array to another.

**Nota**

this will create a new array with the same capacity and size as the source array.

**Parametri**

- **target** -- pointer to an `lv_array_t` variable to copy to
- **source** -- pointer to an `lv_array_t` variable to copy from

**static inline void lv\_array\_clear(lv\_array\_t \*array)**

Remove all elements in array.

**Parametri**

**array** -- pointer to an `lv_array_t` variable

**void lv\_array\_shrink(lv\_array\_t \*array)**

Shrink the memory capacity of array if necessary.

**Parametri**

**array** -- pointer to an `lv_array_t` variable

**lv\_result\_t lv\_array\_remove(lv\_array\_t \*array, uint32\_t index)**

Remove the element at the specified position in the array.

**Parametri**

- **array** -- pointer to an `lv_array_t` variable
- **index** -- the index of the element to remove

**Ritorna**

LV\_RESULT\_OK: success, otherwise: error

**lv\_result\_t lv\_array\_erase(lv\_array\_t \*array, uint32\_t start, uint32\_t end)**

Remove from the array either a single element or a range of elements ([start, end)).

**Nota**

This effectively reduces the container size by the number of elements removed.

**Nota**

When start equals to end, the function has no effect.

**Parametri**

- **array** -- pointer to an `lv_array_t` variable
- **start** -- the index of the first element to be removed
- **end** -- the index of the first element that is not to be removed

**Ritorna**

LV\_RESULT\_OK: success, otherwise: error

**lv\_result\_t lv\_array\_concat(lv\_array\_t \*array, const lv\_array\_t \*other)**

Concatenate two arrays. Adds new elements to the end of the array.

**Nota**

The destination array is automatically expanded as necessary.

**Parametri**

- **array** -- pointer to an `lv_array_t` variable
- **other** -- pointer to the array to concatenate

**Ritorna**

`LV_RESULT_OK`: success, otherwise: error

`lv_result_t lv_array_push_back(lv_array_t *array, const void *element)`

Push back element. Adds a new element to the end of the array. If the array capacity is not enough for the new element, the array will be resized automatically.

 **Nota**

If the element is `NULL`, it will be added as an empty element.

**Parametri**

- **array** -- pointer to an `lv_array_t` variable
- **element** -- pointer to the element to add. `NULL` to push an empty element.

**Ritorna**

`LV_RESULT_OK`: success, otherwise: error

`lv_result_t lv_array_assign(lv_array_t *array, uint32_t index, const void *value)`

Assigns one content to the array, replacing its current content.

**Parametri**

- **array** -- pointer to an `lv_array_t` variable
- **index** -- the index of the element to replace
- **value** -- pointer to the elements to add

**Ritorna**

true: success; false: error

`void *lv_array_at(const lv_array_t *array, uint32_t index)`

Returns a pointer to the element at position `n` in the array.

**Parametri**

- **array** -- pointer to an `lv_array_t` variable
- **index** -- the index of the element to return

**Ritorna**

a pointer to the requested element, `NULL` if `index` is out of range

`static inline void *lv_array_front(const lv_array_t *array)`

Returns a pointer to the first element in the array.

**Parametri**

- **array** -- pointer to an `lv_array_t` variable

**Ritorna**

a pointer to the first element in the array

`static inline void *lv_array_back(const lv_array_t *array)`

Returns a pointer to the last element in the array.

**Parametri**

- **array** -- pointer to an `lv_array_t` variable

`struct _lv_array_t`

`#include <lv_array.h>` Description of a array

**Public Members**

`uint8_t *data`

`uint32_t size`

```
uint32_t capacity  
uint32_t element_size  
bool inner_alloc
```

## 14.18.8 lv\_assert.h

### Defines

```
LV_ASSERT(expr)  
LV_ASSERT_MSG(expr, msg)  
LV_ASSERT_FORMAT_MSG(expr, format, ...)  
LV_ASSERT_NULL(p)  
LV_ASSERT_MALLOC(p)  
LV_ASSERT_MEM_INTEGRITY()
```

### Functions

```
void lv_assert_handler(void)
```

## 14.18.9 lv\_async.h

### Typedef

```
typedef void (*lv_async_cb_t)(void*)
```

Type for async callback.

### Functions

```
lv_result_t lv_async_call(lv_async_cb_t async_xcb, void *user_data)
```

Call an asynchronous function the next time [\*lv\\_timer\\_handler\(\)\*](#) is run. This function is likely to return **before** the call actually happens!

#### Parametri

- **async\_xcb** -- a callback which is the task itself. (the 'x' in the argument name indicates that it's not a fully generic function because it not follows the `func_name(object, callback, ...)` convention)
- **user\_data** -- custom parameter

```
lv_result_t lv_async_call_cancel(lv_async_cb_t async_xcb, void *user_data)
```

Cancel an asynchronous function call

#### Parametri

- **async\_xcb** -- a callback which is the task itself.
- **user\_data** -- custom parameter

## 14.18.10 lv bidi.h

### Defines

```
LV_BIDI_LRO
```

Special non printable strong characters. They can be inserted to texts to affect the run's direction

## LV\_BIDI\_RLO

### Enums

enum **lv\_base\_dir\_t**

*Values:*

enumerator **LV\_BASE\_DIR\_LTR**

enumerator **LV\_BASE\_DIRRTL**

enumerator **LV\_BASE\_DIR\_AUTO**

enumerator **LV\_BASE\_DIR\_NEUTRAL**

enumerator **LV\_BASE\_DIR\_WEAK**

### Functions

void **lv\_bidi\_calculate\_align**(*lv\_text\_align\_t* \*align, *lv\_base\_dir\_t* \*base\_dir, const char \*txt)

Get the real text alignment from the a text alignment, base direction and a text.

**Parametri**

- **align** -- LV\_TEXT\_ALIGN\_..., write back the calculated align here (LV\_TEXT\_ALIGN\_LEFT/RIGHT/CENTER)
- **base\_dir** -- LV\_BASE\_DIR\_..., write the calculated base dir here (LV\_BASE\_DIR\_LTR/RTL)
- **txt** -- a text, used with LV\_BASE\_DIR\_AUTO to determine the base direction

void **lv\_bidi\_set\_custom\_neutrals\_static**(const char \*neutrals)

Set custom neutrals string

**Parametri**

**neutrals** -- default "\t\n\r.,;`\"!%&\r-=(\r){ }<>@#&\$!"

## 14.18.11 lv\_bidi\_private.h

### Functions

void **lv\_bidi\_process**(const char \*str\_in, char \*str\_out, *lv\_base\_dir\_t* base\_dir)

Convert a text to get the characters in the correct visual order according to Unicode Bidirectional Algorithm

**Parametri**

- **str\_in** -- the text to process
- **str\_out** -- store the result here. Has the be `strlen(str_in)` length
- **base\_dir** -- LV\_BASE\_DIR\_LTR or LV\_BASE\_DIRRTL

*lv\_base\_dir\_t* **lv\_bidi\_detect\_base\_dir**(const char \*txt)

Auto-detect the direction of a text based on the first strong character

**Parametri**

**txt** -- the text to process

**Ritorna**

LV\_BASE\_DIR\_LTR or LV\_BASE\_DIRRTL

uint16\_t **lv\_bidi\_get\_logical\_pos**(const char \*str\_in, char \*\*bidi\_txt, uint32\_t len, *lv\_base\_dir\_t* base\_dir, uint32\_t visual\_pos, bool \*is\_rtl)

Get the logical position of a character in a line

**Parametri**

- **str\_in** -- the input string. Can be only one line.
- **bidi\_txt** -- internally the text is bidi processed which buffer can be get here. If not required anymore has to freed with `lv_free()` Can be NULL is unused
- **len** -- length of the line in character count
- **base\_dir** -- base direction of the text: `LV_BASE_DIR_LTR` or `LV_BASE_DIR RTL`
- **visual\_pos** -- the visual character position which logical position should be get
- **is\_rtl** -- tell the char at `visual_pos` is RTL or LTR context

**Ritorna**

the logical character position

```
uint16_t lv_bidi_get_visual_pos(const char *str_in, char **bidi_txt, uint16_t len, lv_base_dir_t
                                base_dir, uint32_t logical_pos, bool *is_rtl)
```

Get the visual position of a character in a line

**Parametri**

- **str\_in** -- the input string. Can be only one line.
- **bidi\_txt** -- internally the text is bidi processed which buffer can be get here. If not required anymore has to freed with `lv_free()` Can be NULL is unused
- **len** -- length of the line in character count
- **base\_dir** -- base direction of the text: `LV_BASE_DIR_LTR` or `LV_BASE_DIR RTL`
- **logical\_pos** -- the logical character position which visual position should be get
- **is\_rtl** -- tell the char at `logical_pos` is RTL or LTR context

**Ritorna**

the visual character position

```
void lv_bidi_process_paragraph(const char *str_in, char *str_out, uint32_t len, lv_base_dir_t base_dir,
                               uint16_t *pos_conv_out, uint16_t pos_conv_len)
```

Bidi process a paragraph of text

**Parametri**

- **str\_in** -- the string to process
- **str\_out** -- store the result here
- **len** -- length of the text
- **base\_dir** -- base dir of the text
- **pos\_conv\_out** -- an `uint16_t` array to store the related logical position of the character. Can be NULL is unused
- **pos\_conv\_len** -- length of `pos_conv_out` in element count

## 14.18.12 lv\_circle\_buf.h

### TypeDef

```
typedef bool (*lv_circle_buf_fill_cb_t)(void *buf, uint32_t buff_len, int32_t index, void *user_data)
```

### Functions

```
lv_circle_buf_t *lv_circle_buf_create(uint32_t capacity, uint32_t element_size)
```

Create a circle buffer

**Parametri**

- **capacity** -- the maximum number of elements in the buffer
- **element\_size** -- the size of an element in bytes

**Ritorna**

pointer to the created buffer

```
lv_circle_buf_t *lv_circle_buf_create_from_buf(void *buf, uint32_t capacity, uint32_t
                                              element_size)
```

Create a circle buffer from an existing buffer

**Parametri**

- **buf** -- pointer to a buffer
- **capacity** -- the maximum number of elements in the buffer
- **element\_size** -- the size of an element in bytes

**Ritorna**

pointer to the created buffer

**lv\_circle\_buf\_t \*lv\_circle\_buf\_create\_from\_array(const lv\_array\_t \*array)**

Create a circle buffer from an existing array

**Parametri**

**array** -- pointer to an array

**Ritorna**

pointer to the created buffer

**lv\_result\_t lv\_circle\_buf\_resize(lv\_circle\_buf\_t \*circle\_buf, uint32\_t capacity)**

Resize the buffer

**Parametri**

- **circle\_buf** -- pointer to a buffer
- **capacity** -- the new capacity of the buffer

**Ritorna**

LV\_RESULT\_OK: the buffer is resized; LV\_RESULT\_INVALID: the buffer is not resized

**void lv\_circle\_buf\_destroy(lv\_circle\_buf\_t \*circle\_buf)**

Destroy a circle buffer

**Parametri**

**circle\_buf** -- pointer to buffer

**uint32\_t lv\_circle\_buf\_size(const lv\_circle\_buf\_t \*circle\_buf)**

Get the size of the buffer

**Parametri**

**circle\_buf** -- pointer to buffer

**Ritorna**

the number of elements in the buffer

**uint32\_t lv\_circle\_buf\_capacity(const lv\_circle\_buf\_t \*circle\_buf)**

Get the capacity of the buffer

**Parametri**

**circle\_buf** -- pointer to buffer

**Ritorna**

the maximum number of elements in the buffer

**uint32\_t lv\_circle\_buf\_remain(const lv\_circle\_buf\_t \*circle\_buf)**

Get the remaining space in the buffer

**Parametri**

**circle\_buf** -- pointer to buffer

**Ritorna**

the number of elements that can be written to the buffer

**bool lv\_circle\_buf\_is\_empty(const lv\_circle\_buf\_t \*circle\_buf)**

Check if the buffer is empty

**Parametri**

**circle\_buf** -- pointer to buffer

**Ritorna**

true: the buffer is empty; false: the buffer is not empty

**bool lv\_circle\_buf\_is\_full(const lv\_circle\_buf\_t \*circle\_buf)**

Check if the buffer is full

**Parametri**

**circle\_buf** -- pointer to buffer

**Ritorna**

true: the buffer is full; false: the buffer is not full

```
void lv_circle_buf_reset(lv_circle_buf_t *circle_buf)
    Reset the buffer
    Parametri
        circle_buf -- pointer to buffer
    Ritorna
        LV_RESULT_OK: the buffer is reset; LV_RESULT_INVALID: the buffer is not reset

void *lv_circle_buf_head(const lv_circle_buf_t *circle_buf)
    Get the head of the buffer
    Parametri
        circle_buf -- pointer to buffer
    Ritorna
        pointer to the head of the buffer

void *lv_circle_buf_tail(const lv_circle_buf_t *circle_buf)
    Get the tail of the buffer
    Parametri
        circle_buf -- pointer to buffer
    Ritorna
        pointer to the tail of the buffer

lv_result_t lv_circle_buf_read(lv_circle_buf_t *circle_buf, void *data)
    Read a value
    Parametri
        • circle_buf -- pointer to buffer
        • data -- pointer to a variable to store the read value
    Ritorna
        LV_RESULT_OK: the value is read; LV_RESULT_INVALID: the value is not read

lv_result_t lv_circle_buf_write(lv_circle_buf_t *circle_buf, const void *data)
    Write a value
    Parametri
        • circle_buf -- pointer to buffer
        • data -- pointer to the value to write
    Ritorna
        LV_RESULT_OK: the value is written; LV_RESULT_INVALID: the value is not written

uint32_t lv_circle_buf_fill(lv_circle_buf_t *circle_buf, uint32_t count, lv_circle_buf_fill_cb_t fill_cb,
                           void *user_data)
    Fill the buffer with values
    Parametri
        • circle_buf -- pointer to buffer
        • count -- the number of values to fill
        • fill_cb -- the callback function to fill the buffer
        • user_data --
    Ritorna
        the number of values filled

lv_result_t lv_circle_buf_skip(lv_circle_buf_t *circle_buf)
    Skip a value
    Parametri
        circle_buf -- pointer to buffer
    Ritorna
        LV_RESULT_OK: the value is skipped; LV_RESULT_INVALID: the value is not skipped

lv_result_t lv_circle_buf_peek(const lv_circle_buf_t *circle_buf, void *data)
    Peek a value
    Parametri
        • circle_buf -- pointer to buffer
        • data -- pointer to a variable to store the peeked value
```

**Ritorna**

LV\_RESULT\_OK: the value is peeked; LV\_RESULT\_INVALID: the value is not peeked

`lv_result_t lv_circle_buf_peek_at(const lv_circle_buf_t *circle_buf, uint32_t index, void *data)`

Peek a value at an index

**Parametri**

- **circle\_buf** -- pointer to buffer
- **index** -- the index of the value to peek, if the index is greater than the size of the buffer, it will return loopy.
- **data** -- pointer to a variable to store the peeked value

**Ritorna**

LV\_RESULT\_OK: the value is peeked; LV\_RESULT\_INVALID: the value is not peeked

### 14.18.13 lv\_color.h

#### Defines

**LV\_COLOR\_NATIVE\_WITH\_ALPHA\_SIZE**

**LV\_OPA\_MIN**

Fully transparent if opa <= LV\_OPA\_MIN

**LV\_OPA\_MAX**

Fully cover if opa >= LV\_OPA\_MAX

**LV\_COLOR\_FORMAT\_GET\_BPP(cf)**

Get the pixel size of a color format in bits, bpp

↻ **Vedi anche**

[lv\\_color\\_format\\_get\\_bpp](#)

**Parametri**

- **cf** -- a color format (LV\_COLOR\_FORMAT\_...)

**Ritorna**

the pixel size in bits

**LV\_COLOR\_FORMAT\_GET\_SIZE(cf)**

Get the pixel size of a color format in bytes

↻ **Vedi anche**

[lv\\_color\\_format\\_get\\_size](#)

**Parametri**

- **cf** -- a color format (LV\_COLOR\_FORMAT\_...)

**Ritorna**

the pixel size in bytes

**LV\_COLOR\_FORMAT\_IS\_ALPHA\_ONLY(cf)**

**LV\_COLOR\_FORMAT\_IS\_INDEXED(cf)**

**LV\_COLOR\_FORMAT\_IS\_YUV(cf)**

**LV\_COLOR\_INDEXED\_PALETTE\_SIZE(cf)**

**LV\_COLOR\_MAKE**(r8, g8, b8)

**LV\_OPA\_MIX2**(a1, a2)

**LV\_OPA\_MIX3**(a1, a2, a3)

## Enums

enum **lv\_opacity\_level\_t**

Opacity percentages.

*Values:*

enumerator **LV\_OPA\_TRANSPIRANT**

enumerator **LV\_OPA\_0**

enumerator **LV\_OPA\_10**

enumerator **LV\_OPA\_20**

enumerator **LV\_OPA\_30**

enumerator **LV\_OPA\_40**

enumerator **LV\_OPA\_50**

enumerator **LV\_OPA\_60**

enumerator **LV\_OPA\_70**

enumerator **LV\_OPA\_80**

enumerator **LV\_OPA\_90**

enumerator **LV\_OPA\_100**

enumerator **LV\_OPA\_COVER**

enum **lv\_color\_format\_t**

*Values:*

enumerator **LV\_COLOR\_FORMAT\_UNKNOWN**

enumerator **LV\_COLOR\_FORMAT\_RAW**

enumerator **LV\_COLOR\_FORMAT\_RAW\_ALPHA**

enumerator **LV\_COLOR\_FORMAT\_L8**

enumerator **LV\_COLOR\_FORMAT\_I1**

enumerator **LV\_COLOR\_FORMAT\_I2**

enumerator **LV\_COLOR\_FORMAT\_I4**

enumerator **LV\_COLOR\_FORMAT\_I8**

enumerator **LV\_COLOR\_FORMAT\_A8**

enumerator **LV\_COLOR\_FORMAT\_RGB565**

enumerator **LV\_COLOR\_FORMAT\_ARGB8565**

Not supported by sw renderer yet.

enumerator **LV\_COLOR\_FORMAT\_RGB565A8**

Color array followed by Alpha array

enumerator **LV\_COLOR\_FORMAT\_AL88**

L8 with alpha >

enumerator **LV\_COLOR\_FORMAT\_RGB565\_SWAPPED**

enumerator **LV\_COLOR\_FORMAT\_RGB888**

enumerator **LV\_COLOR\_FORMAT\_ARGB8888**

enumerator **LV\_COLOR\_FORMAT\_XRGB8888**

enumerator **LV\_COLOR\_FORMAT\_ARGB8888\_PREMULTIPLIED**

enumerator **LV\_COLOR\_FORMAT\_A1**

enumerator **LV\_COLOR\_FORMAT\_A2**

enumerator **LV\_COLOR\_FORMAT\_A4**

enumerator **LV\_COLOR\_FORMAT\_ARGB1555**

enumerator **LV\_COLOR\_FORMAT\_ARGB4444**

enumerator **LV\_COLOR\_FORMAT\_ARGB2222**

enumerator **LV\_COLOR\_FORMAT\_YUV\_START**

enumerator **LV\_COLOR\_FORMAT\_I420**

enumerator **LV\_COLOR\_FORMAT\_I422**

enumerator **LV\_COLOR\_FORMAT\_I444**

enumerator **LV\_COLOR\_FORMAT\_I400**

enumerator **LV\_COLOR\_FORMAT\_NV21**

enumerator **LV\_COLOR\_FORMAT\_NV12**

enumerator **LV\_COLOR\_FORMAT\_YUY2**

enumerator **LV\_COLOR\_FORMAT\_UYVY**

enumerator **LV\_COLOR\_FORMAT\_YUV\_END**

enumerator **LV\_COLOR\_FORMAT\_PROPRIETARY\_START**

enumerator **LV\_COLOR\_FORMAT\_NEMA\_TSC\_START**

enumerator **LV\_COLOR\_FORMAT\_NEMA\_TSC4**

enumerator **LV\_COLOR\_FORMAT\_NEMA\_TSC6**

enumerator **LV\_COLOR\_FORMAT\_NEMA\_TSC6A**

enumerator **LV\_COLOR\_FORMAT\_NEMA\_TSC6AP**

enumerator **LV\_COLOR\_FORMAT\_NEMA\_TSC12**

enumerator **LV\_COLOR\_FORMAT\_NEMA\_TSC12A**

enumerator **LV\_COLOR\_FORMAT\_NEMA\_TSC\_END**

enumerator **LV\_COLOR\_FORMAT\_NATIVE**

enumerator **LV\_COLOR\_FORMAT\_NATIVE\_WITH\_ALPHA**

## Functions

`uint8_t lv_color_format_get_bpp(lv_color_format_t cf)`

Get the pixel size of a color format in bits, bpp

 **Vedi anche**

[LV\\_COLOR\\_FORMAT\\_GET\\_BPP](#)

**Parametri**

**cf** -- a color format (LV\_COLOR\_FORMAT\_...)

**Ritorna**

the pixel size in bits

uint8\_t **lv\_color\_format\_get\_size**(*lv\_color\_format\_t* cf)

Get the pixel size of a color format in bytes

 **Vedi anche**

[LV\\_COLOR\\_FORMAT\\_GET\\_SIZE](#)

**Parametri**

**cf** -- a color format (LV\_COLOR\_FORMAT\_...)

**Ritorna**

the pixel size in bytes

bool **lv\_color\_format\_has\_alpha**(*lv\_color\_format\_t* src\_cf)

Check if a color format has alpha channel or not

**Parametri**

**src\_cf** -- a color format (LV\_COLOR\_FORMAT\_...)

**Ritorna**

true: has alpha channel; false: doesn't have alpha channel

*lv\_color32\_t* **lv\_color\_to\_32**(*lv\_color\_t* color, *lv\_opa\_t* opa)

Create an ARGB8888 color from RGB888 + alpha

**Parametri**

- **color** -- an RGB888 color
- **opa** -- the alpha value

**Ritorna**

the ARGB8888 color

uint32\_t **lv\_color\_to\_int**(*lv\_color\_t* c)

Convert an RGB888 color to an integer

**Parametri**

**c** -- an RGB888 color

**Ritorna**

C as an integer

bool **lv\_color\_eq**(*lv\_color\_t* c1, *lv\_color\_t* c2)

Check if two RGB888 color are equal

**Parametri**

- **c1** -- the first color
- **c2** -- the second color

**Ritorna**

true: equal

bool **lv\_color32\_eq**(*lv\_color32\_t* c1, *lv\_color32\_t* c2)

Check if two ARGB8888 color are equal

**Parametri**

- **c1** -- the first color
- **c2** -- the second color

**Ritorna**

true: equal

***lv\_color\_t* **lv\_color\_hex**(*uint32\_t c*)**

Create a color from 0x000000..0xffff input

**Parametri**

- **c** -- the hex input

**Ritorna**

the color

***lv\_color\_t* **lv\_color\_make**(*uint8\_t r, uint8\_t g, uint8\_t b*)**

Create an RGB888 color

**Parametri**

- **r** -- the red channel (0..255)
- **g** -- the green channel (0..255)
- **b** -- the blue channel (0..255)

**Ritorna**

the color

***lv\_color32\_t* **lv\_color32\_make**(*uint8\_t r, uint8\_t g, uint8\_t b, uint8\_t a*)**

Create an ARGB8888 color

**Parametri**

- **r** -- the red channel (0..255)
- **g** -- the green channel (0..255)
- **b** -- the blue channel (0..255)
- **a** -- the alpha channel (0..255)

**Ritorna**

the color

***lv\_color\_t* **lv\_color\_hex3**(*uint32\_t c*)**

Create a color from 0x000..0xffff input

**Parametri**

- **c** -- the hex input (e.g. 0x123 will be 0x112233)

**Ritorna**

the color

***uint16\_t* **lv\_color\_to\_u16**(*lv\_color\_t* color)**Convert am RGB888 color to RGB565 stored in *uint16\_t***Parametri**

- **color** -- and RGB888 color

**Ritorna**color as RGB565 on *uint16\_t****uint32\_t* **lv\_color\_to\_u32**(*lv\_color\_t* color)**Convert am RGB888 color to XRGB8888 stored in *uint32\_t***Parametri**

- **color** -- and RGB888 color

**Ritorna**color as XRGB8888 on *uint32\_t* (the alpha channel is always set to 0xFF)***uint16\_t* **lv\_color\_16\_16\_mix**(*uint16\_t c1, uint16\_t c2, uint8\_t mix*)**

Mix two RGB565 colors

**Parametri**

- **c1** -- the first color (typically the foreground color)
- **c2** -- the second color (typically the background color)
- **mix** -- 0..255, or LV\_OPA\_0/10/20...

**Ritorna**

mix == 0: c2 mix == 255: c1 mix == 128: 0.5 x c1 + 0.5 x c2

***lv\_color\_t* **lv\_color\_lighten**(*lv\_color\_t* c, *lv\_opa\_t* lvl)**

Mix white to a color

**Parametri**

- **c** -- the base color

- **lvl** -- the intensity of white (0: no change, 255: fully white)

**Ritorna**

the mixed color

***lv\_color\_t lv\_color\_darken(lv\_color\_t c, lv\_opa\_t lvl)***

Mix black to a color

**Parametri**

- **c** -- the base color
- **lvl** -- the intensity of black (0: no change, 255: fully black)

**Ritorna**

the mixed color

***lv\_color\_t lv\_color\_hsv\_to\_rgb(uint16\_t h, uint8\_t s, uint8\_t v)***

Convert a HSV color to RGB

**Parametri**

- **h** -- hue [0..359]
- **s** -- saturation [0..100]
- **v** -- value [0..100]

**Ritorna**

the given RGB color in RGB (with LV\_COLOR\_DEPTH depth)

***lv\_color\_hsv\_t lv\_color\_rgb\_to\_hsv(uint8\_t r8, uint8\_t g8, uint8\_t b8)***

Convert a 32-bit RGB color to HSV

**Parametri**

- **r8** -- 8-bit red
- **g8** -- 8-bit green
- **b8** -- 8-bit blue

**Ritorna**

the given RGB color in HSV

***lv\_color\_hsv\_t lv\_color\_to\_hsv(lv\_color\_t color)***

Convert a color to HSV

**Parametri****color** -- color**Ritorna**

the given color in HSV

***lv\_color\_t lv\_color\_white(void)***

A helper for white color

**Ritorna**

a white color

***lv\_color\_t lv\_color\_black(void)***

A helper for black color

**Ritorna**

a black color

***void lv\_color\_premultiply(lv\_color32\_t \*c)******void lv\_color16\_premultiply(lv\_color16\_t \*c, lv\_opa\_t a)******uint8\_t lv\_color\_luminance(lv\_color\_t c)***

Get the luminance of a color: luminance = 0.3 R + 0.59 G + 0.11 B

**Parametri****c** -- a color**Ritorna**

the brightness [0..255]

***uint8\_t lv\_color16\_luminance(const lv\_color16\_t c)***

Get the luminance of a color16: luminance = 0.3 R + 0.59 G + 0.11 B

**Parametri****C** -- a color**Ritorna**

the brightness [0..255]

**uint8\_t lv\_color24\_luminance(const uint8\_t \*c)**

Get the luminance of a color24: luminance = 0.3 R + 0.59 G + 0.11 B

**Parametri****C** -- a color**Ritorna**

the brightness [0..255]

**uint8\_t lv\_color32\_luminance(*lv\_color32\_t* c)**

Get the luminance of a color32: luminance = 0.3 R + 0.59 G + 0.11 B

**Parametri****C** -- a color**Ritorna**

the brightness [0..255]

**static inline uint16\_t lv\_color\_swap\_16(uint16\_t c)**

Swap the endianness of an rgb565 color

**Parametri****C** -- a color**Ritorna**

the swapped color

**Variables****const lv\_color\_filter\_dsc\_t lv\_color\_filter\_shade****struct lv\_color\_t****Public Members****uint8\_t blue****uint8\_t green****uint8\_t red****struct lv\_color16\_t****Public Members****uint16\_t blue****uint16\_t green****uint16\_t red****struct lv\_color32\_t**

**Public Members**

```
uint8_t blue
```

```
uint8_t green
```

```
uint8_t red
```

```
uint8_t alpha
```

```
struct lv_color_hsv_t
```

**Public Members**

```
uint16_t h
```

```
uint8_t s
```

```
uint8_t v
```

```
struct lv_color16a_t
```

**Public Members**

```
uint8_t lumi
```

```
uint8_t alpha
```

**14.18.14 lv\_color\_op.h****Typedef**

```
typedef lv_color_t (*lv_color_filter_cb_t)(const struct lv_color_filter_dsc_t*, lv_color_t, lv_opa_t)
```

**Functions**

*lv\_color\_t* **lv\_color\_mix**(*lv\_color\_t* c1, *lv\_color\_t* c2, uint8\_t mix)

Mix two colors with a given ratio.

**Parametri**

- **c1** -- the first color to mix (usually the foreground)
- **c2** -- the second color to mix (usually the background)
- **mix** -- The ratio of the colors. 0: full c2, 255: full c1, 127: half c1 and half c2

**Ritorna**

the mixed color

*lv\_color32\_t* **lv\_color\_mix32**(*lv\_color32\_t* fg, *lv\_color32\_t* bg)

 **Nota**

Use bg.alpha in the return value

**Nota**

Use `fg.alpha` as mix ratio

**Parametri**

- **fg** --
- **bg** --

**Ritorna**

`lv_color32_t lv_color_mix32_premultiplied(lv_color32_t fg, lv_color32_t bg)`

Blends two premultiplied ARGB8888 colors while maintaining correct alpha compositing.

This function correctly blends the foreground (fg) and background (bg) colors, ensuring that the output remains in a premultiplied alpha format.

**Nota**

If the foreground is fully opaque, it is returned as is.

**Nota**

If the foreground is fully transparent, the background is returned.

**Parametri**

- **fg** -- The foreground color in premultiplied ARGB8888 format.
- **bg** -- The background color in premultiplied ARGB8888 format.

**Ritorna**

The resulting blended color in premultiplied ARGB8888 format.

`uint8_t lv_color_brightness(lv_color_t c)`

Get the brightness of a color

**Parametri**

- **c** -- a color

**Ritorna**

brightness in range [0..255]

`void lv_color_filter_dsc_init(lv_color_filter_dsc_t *dsc, lv_color_filter_cb_t cb)`

`lv_color32_t lv_color_over32(lv_color32_t fg, lv_color32_t bg)`

Blend two colors that have not been pre-multiplied using their alpha values

**Parametri**

- **fg** -- the foreground color
- **bg** -- the background color

**Ritorna**

result color

`struct _lv_color_filter_dsc_t`

**Public Members**

`lv_color_filter_cb_t filter_cb`

`void *user_data`

**14.18.15 lv\_color\_op\_private.h****14.18.16 lv\_event.h****Typedef**

```
typedef void (*lv_event_cb_t)(lv_event_t *e)
```

**Enums****enum lv\_event\_code\_t**

Type of event being sent to Widget

*Values:*

enumerator **LV\_EVENT\_ALL**

enumerator **LV\_EVENT\_PRESSED**

Input device events Widget has been pressed

enumerator **LV\_EVENT\_PRESSING**

Widget is being pressed (sent continuously while pressing)

enumerator **LV\_EVENT\_PRESS\_LOST**

Widget is still being pressed but slid cursor/finger off Widget

enumerator **LV\_EVENT\_SHORT\_CLICKED**

Widget was pressed for a short period of time, then released. Not sent if scrolled.

enumerator **LV\_EVENT\_SINGLE\_CLICKED**

Sent for first short click within a small distance and short time

enumerator **LV\_EVENT\_DOUBLE\_CLICKED**

Sent for second short click within small distance and short time

enumerator **LV\_EVENT\_TRIPLE\_CLICKED**

Sent for third short click within small distance and short time

enumerator **LV\_EVENT\_LONG\_PRESSED**

Object has been pressed for at least `long_press_time`. Not sent if scrolled.

enumerator **LV\_EVENT\_LONG\_PRESSED\_REPEAT**

Sent after `long_press_time` in every `long_press_repeat_time` ms. Not sent if scrolled.

enumerator **LV\_EVENT\_CLICKED**

Sent on release if not scrolled (regardless to long press)

enumerator **LV\_EVENT\_RELEASED**

Sent in every cases when Widget has been released

**enumerator LV\_EVENT\_SCROLL\_BEGIN**

Scrolling begins. The event parameter is a pointer to the animation of the scroll. Can be modified

**enumerator LV\_EVENT\_SCROLL\_THROW\_BEGIN****enumerator LV\_EVENT\_SCROLL\_END**

Scrolling ends

**enumerator LV\_EVENT\_SCROLL**

Scrolling

**enumerator LV\_EVENT\_GESTURE**

A gesture is detected. Get gesture with `lv_indev_get_gesture_dir(lv_indev_active());`

**enumerator LV\_EVENT\_KEY**

A key is sent to Widget. Get key with `lv_indev_get_key(lv_indev_active());`

**enumerator LV\_EVENT\_ROTARY**

An encoder or wheel was rotated. Get rotation count with `lv_event_get_rotary_diff(e);`

**enumerator LV\_EVENT\_FOCUSED**

Widget received focus

**enumerator LV\_EVENT\_DEFOCUSSED**

Widget's focus has been lost

**enumerator LV\_EVENT\_LEAVE**

Widget's focus has been lost but is still selected

**enumerator LV\_EVENT\_HIT\_TEST**

Perform advanced hit-testing

**enumerator LV\_EVENT\_INDEV\_RESET**

Indev has been reset

**enumerator LV\_EVENT\_HOVER\_OVER**

Indev hover over object

**enumerator LV\_EVENT\_HOVER\_LEAVE**

Indev hover leave object

**enumerator LV\_EVENT\_COVER\_CHECK**

Drawing events Check if Widget fully covers an area. The event parameter is `lv_cover_check_info_t *`.

**enumerator LV\_EVENT\_REFR\_EXT\_DRAW\_SIZE**

Get required extra draw area around Widget (e.g. for shadow). The event parameter is `int32_t *` to store the size.

**enumerator LV\_EVENT\_DRAW\_MAIN\_BEGIN**

Starting the main drawing phase

**enumerator LV\_EVENT\_DRAW\_MAIN**

Perform the main drawing

**enumerator LV\_EVENT\_DRAW\_MAIN\_END**

Finishing the main drawing phase

**enumerator LV\_EVENT\_DRAW\_POST\_BEGIN**

Starting the post draw phase (when all children are drawn)

**enumerator LV\_EVENT\_DRAW\_POST**

Perform the post draw phase (when all children are drawn)

**enumerator LV\_EVENT\_DRAW\_POST\_END**

Finishing the post draw phase (when all children are drawn)

**enumerator LV\_EVENT\_DRAW\_TASK\_ADDED**

Adding a draw task. The `LV_OBJ_FLAG_SEND_DRAW_TASK_EVENTS` flag needs to be set

**enumerator LV\_EVENT\_VALUE\_CHANGED**

Special events Widget's value has changed (i.e. slider moved)

**enumerator LV\_EVENT\_INSERT**

Text has been inserted into Widget. The event data is `char *` being inserted.

**enumerator LV\_EVENT\_REFRESH**

Notify Widget to refresh something on it (for user)

**enumerator LV\_EVENT\_READY**

A process has finished

**enumerator LV\_EVENT\_CANCEL**

A process has been cancelled

**enumerator LV\_EVENT\_CREATE**

Other events Object is being created

**enumerator LV\_EVENT\_DELETE**

Object is being deleted

**enumerator LV\_EVENT\_CHILD\_CHANGED**

Child was removed, added, or its size, position were changed

**enumerator LV\_EVENT\_CHILD\_CREATED**

Child was created, always bubbles up to all parents

**enumerator LV\_EVENT\_CHILD\_DELETED**

Child was deleted, always bubbles up to all parents

**enumerator LV\_EVENT\_SCREEN\_UNLOAD\_START**

A screen unload started, fired immediately when scr\_load is called

**enumerator LV\_EVENT\_SCREEN\_LOAD\_START**

A screen load started, fired when the screen change delay is expired

**enumerator LV\_EVENT\_SCREEN\_LOADED**

A screen was loaded

**enumerator LV\_EVENT\_SCREEN\_UNLOADED**

A screen was unloaded

**enumerator LV\_EVENT\_SIZE\_CHANGED**

Object coordinates/size have changed

**enumerator LV\_EVENT\_STYLE\_CHANGED**

Object's style has changed

**enumerator LV\_EVENT\_LAYOUT\_CHANGED**

A child's position position has changed due to a layout recalculation

**enumerator LV\_EVENT\_GET\_SELF\_SIZE**

Get internal size of a widget

**enumerator LV\_EVENT\_INVALIDATE\_AREA**

Events of optional LVGL components An area is invalidated (marked for redraw). `lv_event_get_param(e)` returns a pointer to an `lv_area_t` object with the coordinates of the area to be invalidated. The area can be freely modified if needed to adapt it a special requirement of the display. Usually needed with monochrome displays to invalidate  $N \times 8$  rows or columns in one pass.

**enumerator LV\_EVENT\_RESOLUTION\_CHANGED**

Sent when the resolution changes due to `lv_display_set_resolution()` or `lv_display_set_rotation()`.

**enumerator LV\_EVENT\_COLOR\_FORMAT\_CHANGED**

Sent as a result of any call to `lv_display_set_color_format()`.

**enumerator LV\_EVENT\_REFR\_REQUEST**

Sent when something happened that requires redraw.

**enumerator LV\_EVENT\_REFR\_START**

Sent before a refreshing cycle starts. Sent even if there is nothing to redraw.

**enumerator LV\_EVENT\_REFR\_READY**

Sent when refreshing has been completed (after rendering and calling flush callback). Sent even if no redraw happened.

enumerator **LV\_EVENT\_RENDER\_START**

Sent just before rendering begins.

enumerator **LV\_EVENT\_RENDER\_READY**

Sent after rendering has been completed (before calling flush callback)

enumerator **LV\_EVENT\_FLUSH\_START**

Sent before flush callback is called.

enumerator **LV\_EVENT\_FLUSH\_FINISH**

Sent after flush callback call has returned.

enumerator **LV\_EVENT\_FLUSH\_WAIT\_START**

Sent before flush wait callback is called.

enumerator **LV\_EVENT\_FLUSH\_WAIT\_FINISH**

Sent after flush wait callback call has returned.

enumerator **LV\_EVENT\_VSYNC**enumerator **LV\_EVENT\_VSYNC\_REQUEST**enumerator **LV\_EVENT\_LAST**enumerator **LV\_EVENT\_PREPROCESS**

Number of default events

enumerator **LV\_EVENT\_MARKED\_DELETING**

This is a flag that can be set with an event so it's processed before the class default event processing

**Functions**

`lv_result_t lv_event_send(lv_event_list_t *list, lv_event_t *e, bool preprocess)`

Event callback. Events are used to notify the user of some action being taken on Widget. For details, see `::lv_event_t`.

`lv_event_dsc_t *lv_event_add(lv_event_list_t *list, lv_event_cb_t cb, lv_event_code_t filter, void *user_data)`

`bool lv_event_remove_dsc(lv_event_list_t *list, lv_event_dsc_t *dsc)`

`uint32_t lv_event_get_count(lv_event_list_t *list)`

`lv_event_dsc_t *lv_event_get_dsc(lv_event_list_t *list, uint32_t index)`

`lv_event_cb_t lv_event_dsc_get_cb(lv_event_dsc_t *dsc)`

`void *lv_event_dsc_get_user_data(lv_event_dsc_t *dsc)`

`bool lv_event_remove(lv_event_list_t *list, uint32_t index)`

`void lv_event_remove_all(lv_event_list_t *list)`

`void *lv_event_get_target(lv_event_t *e)`

Get Widget originally targeted by the event. It's the same even if event was bubbled.

**Parametri**

`e` -- pointer to the event descriptor

**Ritorna**

the target of the event\_code

`void *lv_event_get_current_target(lv_event_t *e)`

Get current target of the event. It's the Widget for which the event handler being called. If the event is not bubbled it's the same as "normal" target.

**Parametri**

`e` -- pointer to the event descriptor

**Ritorna**

pointer to the current target of the event\_code

`lv_event_code_t lv_event_get_code(lv_event_t *e)`

Get event code of an event.

**Parametri**

`e` -- pointer to the event descriptor

**Ritorna**

the event code. (E.g. LV\_EVENT\_CLICKED, LV\_EVENT\_FOCUSED, etc)

`void *lv_event_get_param(lv_event_t *e)`

Get parameter passed when event was sent.

**Parametri**

`e` -- pointer to the event descriptor

**Ritorna**

pointer to the parameter

`void *lv_event_get_user_data(lv_event_t *e)`

Get user\_data passed when event was registered on Widget.

**Parametri**

`e` -- pointer to the event descriptor

**Ritorna**

pointer to the user\_data

`void lv_event_stop_bubbling(lv_event_t *e)`

Stop event from bubbling. This is only valid when called in the middle of an event processing chain.

**Parametri**

`e` -- pointer to the event descriptor

`void lv_event_stop_trickling(lv_event_t *e)`

Stop event from trickling down to children. This is only valid when called in the middle of an event processing chain.

**Parametri**

`e` -- pointer to the event descriptor

`void lv_event_stop_processing(lv_event_t *e)`

Stop processing this event. This is only valid when called in the middle of an event processing chain.

**Parametri**

`e` -- pointer to the event descriptor

`uint32_t lv_event_register_id(void)`

Register a new, custom event ID. It can be used the same way as e.g. LV\_EVENT\_CLICKED to send custom events

Example:

```
uint32_t LV_EVENT_MINE = 0;
...
e = lv_event_register_id();
...
lv_obj_send_event(obj, LV_EVENT_MINE, &some_data);
```

**Ritorna**  
the new event id

```
const char *lv_event_code_get_name(lv_event_code_t code)
```

Get the name of an event code.

**Parametri**  
**code** -- the event code

**Ritorna**  
the name of the event code as a string

struct **lv\_event\_list\_t**

#### Public Members

**lv\_array\_t array**

**uint8\_t is\_traversing**

True: the list is being nested traversed

**uint8\_t has\_marked\_deleting**

True: the list has marked deleting objects when some of events are marked as deleting

### 14.18.17 lv\_event\_private.h

#### Functions

void **lv\_event\_push**(*lv\_event\_t* \*e)

void **lv\_event\_pop**(*lv\_event\_t* \*e)

void **lv\_event\_mark\_deleted**(*void* \*target)

Nested events can be called and one of them might belong to an object that is being deleted. Mark this object's *event\_temp\_data* deleted to know that its *lv\_obj\_send\_event* should return *LV\_RESULT\_INVALID*

**Parametri**

**target** -- pointer to an event target which was deleted

struct **\_lv\_event\_dsc\_t**

#### Public Members

*lv\_event\_cb\_t* **cb**

**void \*user\_data**

**uint32\_t filter**

struct **\_lv\_event\_t**

#### Public Members

**void \*current\_target**

```
void *original_target
```

```
lv_event_code_t code
```

```
void *user_data
```

```
void *param
```

```
lv_event_t *prev
```

```
uint8_t deleted
```

```
uint8_t stop_processing
```

```
uint8_t stop_bubbling
```

```
uint8_t stop_trickling
```

## 14.18.18 lv\_fs.h

### Defines

**LV\_FS\_MAX\_FN\_LENGTH**

**LV\_FS\_MAX\_PATH\_LENGTH**

**LV\_FS\_CACHE\_FROM\_BUFFER**

### Typedef

typedef struct *lv\_fs\_drv\_t* **lv\_fs\_drv\_t**

### Enums

enum **lv\_fs\_res\_t**

Errors in the file system module.

*Values:*

enumerator **LV\_FS\_RES\_OK**

enumerator **LV\_FS\_RES\_HW\_ERR**

enumerator **LV\_FS\_RES\_FS\_ERR**

enumerator **LV\_FS\_RES\_NOT\_EX**

enumerator **LV\_FS\_RES\_FULL**

enumerator **LV\_FS\_RES\_LOCKED**

enumerator **LV\_FS\_RES\_DENIED**

enumerator **LV\_FS\_RES\_BUSY**

enumerator **LV\_FS\_RES\_TOUT**

enumerator **LV\_FS\_RES\_NOT\_IMP**

enumerator **LV\_FS\_RES\_OUT\_OF\_MEM**

enumerator **LV\_FS\_RES\_INV\_PARAM**

enumerator **LV\_FS\_RES\_UNKNOWN**

enum **lv\_fs\_mode\_t**

File open mode.

*Values:*

enumerator **LV\_FS\_MODE\_WR**

enumerator **LV\_FS\_MODE\_RD**

enum **lv\_fs\_whence\_t**

Seek modes.

*Values:*

enumerator **LV\_FS\_SEEK\_SET**

Set the position from absolutely (from the start of file)

enumerator **LV\_FS\_SEEK\_CUR**

Set the position from the current position

enumerator **LV\_FS\_SEEK\_END**

Set the position from the end of the file

## Functions

void **lv\_fs\_drv\_init**(*lv\_fs\_drv\_t* \*drv)

Initialize a file system driver with default values. It is used to ensure all fields have known values and not memory junk. After it you can set the fields.

### Parametri

**drv** -- pointer to driver variable to initialize

**void lv\_fs\_drv\_register(lv\_fs\_drv\_t \*drv)**

Add a new drive

**Parametri**

- **drv** -- pointer to an lv\_fs\_drv\_t structure which is initied with the corresponding function pointers. Only pointer is saved, so the driver should be static or dynamically allocated.

**lv\_fs\_drv\_t \*lv\_fs\_get\_drv(char letter)**

Give a pointer to a driver from its letter

**Parametri**

- **letter** -- the driver-identifier letter

**Ritorna**

pointer to a driver or NULL if not found

**bool lv\_fs\_is\_ready(char letter)**

Test if a drive is ready or not. If the **ready** function was not initialized **true** will be returned.

**Parametri**

- **letter** -- letter of the drive

**Ritorna**

true: drive is ready; false: drive is not ready

**lv\_fs\_res\_t lv\_fs\_open(lv\_fs\_file\_t \*file\_p, const char \*path, lv\_fs\_mode\_t mode)**

Open a file

**Parametri**

- **file\_p** -- pointer to a *lv\_fs\_file\_t* variable
- **path** -- path to the file beginning with the driver letter (e.g. S:/folder/file.txt)
- **mode** -- read: FS\_MODE\_RD, write: FS\_MODE\_WR, both: FS\_MODE\_RD | FS\_MODE\_WR

**Ritorna**

LV\_FS\_RES\_OK or any error from *lv\_fs\_res\_t* enum

**void lv\_fs\_make\_path\_from\_buffer(lv\_fs\_path\_ex\_t \*path, char letter, const void \*buf, uint32\_t size)**

Make a path object for the memory-mapped file compatible with the file system interface

**Parametri**

- **path** -- path to a *lv\_fs\_path\_ex* object
- **letter** -- the identifier letter of the driver. E.g. LV\_FS\_MEMFS\_LETTER
- **buf** -- address of the memory buffer
- **size** -- size of the memory buffer in bytes

**lv\_fs\_res\_t lv\_fs\_close(lv\_fs\_file\_t \*file\_p)**

Close an already opened file

**Parametri**

- **file\_p** -- pointer to a *lv\_fs\_file\_t* variable

**Ritorna**

LV\_FS\_RES\_OK or any error from *lv\_fs\_res\_t* enum

**lv\_fs\_res\_t lv\_fs\_read(lv\_fs\_file\_t \*file\_p, void \*buf, uint32\_t btr, uint32\_t \*br)**

Read from a file

**Parametri**

- **file\_p** -- pointer to a *lv\_fs\_file\_t* variable
- **buf** -- pointer to a buffer where the read bytes are stored
- **btr** -- Bytes To Read
- **br** -- the number of real read bytes (Bytes Read). NULL if unused.

**Ritorna**

LV\_FS\_RES\_OK or any error from *lv\_fs\_res\_t* enum

**lv\_fs\_res\_t lv\_fs\_write(lv\_fs\_file\_t \*file\_p, const void \*buf, uint32\_t btw, uint32\_t \*bw)**

Write into a file

**Parametri**

- **file\_p** -- pointer to a *lv\_fs\_file\_t* variable
- **buf** -- pointer to a buffer with the bytes to write

- **btw** -- Bytes To Write
- **bw** -- the number of real written bytes (Bytes Written). NULL if unused.

**Ritorna**

LV\_FS\_RES\_OK or any error from *lv\_fs\_res\_t* enum

*lv\_fs\_res\_t* **lv\_fs\_seek**(*lv\_fs\_file\_t* \*file\_p, uint32\_t pos, *lv\_fs\_whence\_t* whence)

Set the position of the 'cursor' (read write pointer) in a file

**Parametri**

- **file\_p** -- pointer to a *lv\_fs\_file\_t* variable
- **pos** -- the new position expressed in bytes index (0: start of file)
- **whence** -- tells from where to set position. See *lv\_fs\_whence\_t*

**Ritorna**

LV\_FS\_RES\_OK or any error from *lv\_fs\_res\_t* enum

*lv\_fs\_res\_t* **lv\_fs\_tell**(*lv\_fs\_file\_t* \*file\_p, uint32\_t \*pos)

Give the position of the read write pointer

**Parametri**

- **file\_p** -- pointer to a *lv\_fs\_file\_t* variable
- **pos** -- pointer to store the position of the read write pointer

**Ritorna**

LV\_FS\_RES\_OK or any error from 'fs\_res\_t'

*lv\_fs\_res\_t* **lv\_fs\_dir\_open**(*lv\_fs\_dir\_t* \*raddir\_p, const char \*path)

Initialize a 'fs\_dir\_t' variable for directory reading

**Parametri**

- **raddir\_p** -- pointer to a '*lv\_fs\_dir\_t*' variable
- **path** -- path to a directory

**Ritorna**

LV\_FS\_RES\_OK or any error from *lv\_fs\_res\_t* enum

*lv\_fs\_res\_t* **lv\_fs\_dir\_read**(*lv\_fs\_dir\_t* \*raddir\_p, char \*fn, uint32\_t fn\_len)

Read the next filename form a directory. The name of the directories will begin with '/'

**Parametri**

- **raddir\_p** -- pointer to an initialized 'fs\_dir\_t' variable
- **fn** -- pointer to a buffer to store the filename
- **fn\_len** -- length of the buffer to store the filename

**Ritorna**

LV\_FS\_RES\_OK or any error from *lv\_fs\_res\_t* enum

*lv\_fs\_res\_t* **lv\_fs\_dir\_close**(*lv\_fs\_dir\_t* \*raddir\_p)

Close the directory reading

**Parametri**

- **raddir\_p** -- pointer to an initialized 'fs\_dir\_t' variable

**Ritorna**

LV\_FS\_RES\_OK or any error from *lv\_fs\_res\_t* enum

char \***lv\_fs\_get\_letters**(char \*buf)

Fill a buffer with the letters of existing drivers

**Parametri**

- **buf** -- buffer to store the letters ('\0' added after the last letter)

**Ritorna**

the buffer

const char \***lv\_fs\_get\_ext**(const char \*fn)

Return with the extension of the filename

**Parametri**

- **fn** -- string with a filename

**Ritorna**

pointer to the beginning extension or empty string if no extension

`char *lv_fs_up(char *path)`

Step up one level

**Parametri**

**path** -- pointer to a file name

**Ritorna**

the truncated file name

`const char *lv_fs_get_last(const char *path)`

Get the last element of a path (e.g. U:/folder/file -> file)

**Parametri**

**path** -- pointer to a file name

**Ritorna**

pointer to the beginning of the last element in the path

`struct _lv_fs_drv_t`

### Public Members

`char letter`

`uint32_t cache_size`

`bool (*ready_cb)(lv_fs_drv_t *drv)`

`void *(*open_cb)(lv_fs_drv_t *drv, const char *path, lv_fs_mode_t mode)`

`lv_fs_res_t (*close_cb)(lv_fs_drv_t *drv, void *file_p)`

`lv_fs_res_t (*read_cb)(lv_fs_drv_t *drv, void *file_p, void *buf, uint32_t btr, uint32_t *br)`

`lv_fs_res_t (*write_cb)(lv_fs_drv_t *drv, void *file_p, const void *buf, uint32_t btw, uint32_t *bw)`

`lv_fs_res_t (*seek_cb)(lv_fs_drv_t *drv, void *file_p, uint32_t pos, lv_fs_whence_t whence)`

`lv_fs_res_t (*tell_cb)(lv_fs_drv_t *drv, void *file_p, uint32_t *pos_p)`

`void *(*dir_open_cb)(lv_fs_drv_t *drv, const char *path)`

`lv_fs_res_t (*dir_read_cb)(lv_fs_drv_t *drv, void *raddir_p, char *fn, uint32_t fn_len)`

`lv_fs_res_t (*dir_close_cb)(lv_fs_drv_t *drv, void *raddir_p)`

`void *user_data`

Custom file user data

`struct lv_fs_file_t`

**Public Members**

```
void *file_d
lv_fs_drv_t *drv
lv_fs_file_cache_t *cache
struct lv_fs_dir_t
```

**Public Members**

```
void *dir_d
```

```
lv_fs_drv_t *drv
```

**14.18.19 lv\_fs\_private.h****Functions**

```
void lv_fs_init(void)
    Initialize the File system interface
void lv_fs_deinit(void)
    Deinitialize the File system interface
```

```
struct _lv_fs_file_cache_t
```

**Public Members**

```
uint32_t start
uint32_t end
uint32_t file_position
void *buffer
```

```
struct _lv_fs_path_ex_t
```

#include <lv\_fs\_private.h> Extended path object to specify buffer for memory-mapped files

**Public Members**

```
char path[4]
    This is needed to make it compatible with a normal path
const void *buffer
uint32_t size
```

## 14.18.20 lv\_grad.h

### Enums

#### enum **lv\_grad\_dir\_t**

The direction of the gradient.

*Values:*

##### enumerator **LV\_GRAD\_DIR\_NONE**

No gradient (the `grad_color` property is ignored)

##### enumerator **LV\_GRAD\_DIR\_VER**

Simple vertical (top to bottom) gradient

##### enumerator **LV\_GRAD\_DIR\_HOR**

Simple horizontal (left to right) gradient

##### enumerator **LV\_GRAD\_DIR\_LINEAR**

Linear gradient defined by start and end points. Can be at any angle.

##### enumerator **LV\_GRAD\_DIR\_RADIAL**

Radial gradient defined by start and end circles

##### enumerator **LV\_GRAD\_DIR\_CONICAL**

Conical gradient defined by center point, start and end angles

#### enum **lv\_grad\_extend\_t**

Gradient behavior outside the defined range.

*Values:*

##### enumerator **LV\_GRAD\_EXTEND\_PAD**

Repeat the same color

##### enumerator **LV\_GRAD\_EXTEND\_REPEAT**

Repeat the pattern

##### enumerator **LV\_GRAD\_EXTEND\_REFLECT**

Repeat the pattern mirrored

### Functions

```
void lv_grad_init_stops(lv_grad_dsc_t *grad, const lv_color_t colors[], const lv_opa_t opa[], const uint8_t fracs[], int num_stops)
```

Initialize gradient color map from a table

#### Parametri

- **grad** -- pointer to a gradient descriptor
- **colors** -- color array
- **fracs** -- position array (0..255): if NULL, then colors are distributed evenly
- **opa** -- opacity array: if NULL, then LV\_OPA\_COVER is assumed
- **num\_stops** -- number of gradient stops (1..LV\_GRADIENT\_MAX\_STOPS)

**void lv\_grad\_horizontal\_init(*lv\_grad\_desc\_t* \*dsc)**

Helper function to initialize a horizontal gradient.

**Parametri**

**dsc** -- gradient descriptor

**void lv\_grad\_vertical\_init(*lv\_grad\_desc\_t* \*dsc)**

Helper function to initialize a vertical gradient.

**Parametri**

**dsc** -- gradient descriptor

**void lv\_grad\_linear\_init(*lv\_grad\_desc\_t* \*dsc, int32\_t from\_x, int32\_t from\_y, int32\_t to\_x, int32\_t to\_y, *lv\_grad\_extend\_t* extend)**

Helper function to initialize linear gradient

**Parametri**

- **dsc** -- gradient descriptor
- **from\_x** -- start x position: can be a coordinate or an *lv\_pct()* value predefined constants LV\_GRAD\_LEFT, LV\_GRAD\_RIGHT, LV\_GRAD\_TOP, LV\_GRAD\_BOTTOM, LV\_GRAD\_CENTER can be used as well
- **from\_y** -- start y position
- **to\_x** -- end x position
- **to\_y** -- end y position
- **extend** -- one of LV\_GRAD\_EXTEND\_PAD, LV\_GRAD\_EXTEND\_REPEAT or LV\_GRAD\_EXTEND\_REFLECT

**void lv\_grad\_radial\_init(*lv\_grad\_desc\_t* \*dsc, int32\_t center\_x, int32\_t center\_y, int32\_t to\_x, int32\_t to\_y, *lv\_grad\_extend\_t* extend)**

Helper function to initialize radial gradient

**Parametri**

- **dsc** -- gradient descriptor
- **center\_x** -- center x position: can be a coordinate or an *lv\_pct()* value predefined constants LV\_GRAD\_LEFT, LV\_GRAD\_RIGHT, LV\_GRAD\_TOP, LV\_GRAD\_BOTTOM, LV\_GRAD\_CENTER can be used as well
- **center\_y** -- center y position
- **to\_x** -- point on the end circle x position
- **to\_y** -- point on the end circle y position
- **extend** -- one of LV\_GRAD\_EXTEND\_PAD, LV\_GRAD\_EXTEND\_REPEAT or LV\_GRAD\_EXTEND\_REFLECT

**void lv\_grad\_radial\_set\_focal(*lv\_grad\_desc\_t* \*dsc, int32\_t center\_x, int32\_t center\_y, int32\_t radius)**

Set focal (starting) circle of a radial gradient

**Parametri**

- **dsc** -- gradient descriptor
- **center\_x** -- center x position: can be a coordinate or an *lv\_pct()* value predefined constants LV\_GRAD\_LEFT, LV\_GRAD\_RIGHT, LV\_GRAD\_TOP, LV\_GRAD\_BOTTOM, LV\_GRAD\_CENTER can be used as well
- **center\_y** -- center y position
- **radius** -- radius of the starting circle (NOTE: this must be a scalar number, not percentage)

**void lv\_grad\_conical\_init(*lv\_grad\_desc\_t* \*dsc, int32\_t center\_x, int32\_t center\_y, int32\_t start\_angle, int32\_t end\_angle, *lv\_grad\_extend\_t* extend)**

Helper function to initialize conical gradient

**Parametri**

- **dsc** -- gradient descriptor
- **center\_x** -- center x position: can be a coordinate or an *lv\_pct()* value predefined constants LV\_GRAD\_LEFT, LV\_GRAD\_RIGHT, LV\_GRAD\_TOP, LV\_GRAD\_BOTTOM, LV\_GRAD\_CENTER can be used as well
- **center\_y** -- center y position
- **start\_angle** -- start angle in degrees

- **end\_angle** -- end angle in degrees
- **extend** -- one of LV\_GRAD\_EXTEND\_PAD, LV\_GRAD\_EXTEND\_REPEAT or LV\_GRAD\_EXTEND\_REFLECT

struct **lv\_grad\_stop\_t**

#include <lv\_grad.h> A gradient stop definition. This matches a color and a position in a virtual 0-255 scale.

#### Public Members

*lv\_color\_t* **color**

The stop color

*lv\_opa\_t* **opa**

The opacity of the color

*uint8\_t* **frac**

The stop position in 1/255 unit

struct **lv\_grad\_dsc\_t**

#include <lv\_grad.h> A descriptor of a gradient.

#### Public Members

*lv\_grad\_stop\_t* **stops[2]**

A gradient stop array

*uint8\_t* **stops\_count**

The number of used stops in the array

*lv\_grad\_dir\_t* **dir**

The gradient direction. Any of LV\_GRAD\_DIR\_NONE, LV\_GRAD\_DIR\_VER, LV\_GRAD\_DIR\_HOR, LV\_GRAD\_TYPE\_LINEAR, LV\_GRAD\_TYPE\_RADIAL, LV\_GRAD\_TYPE\_CONICAL

*lv\_grad\_extend\_t* **extend**

Behaviour outside the defined range. LV\_GRAD\_EXTEND\_NONE, LV\_GRAD\_EXTEND\_PAD, LV\_GRAD\_EXTEND\_REPEAT, LV\_GRAD\_EXTEND\_REFLECT

*lv\_point\_t* **start**

Linear gradient vector start point

*lv\_point\_t* **end**

Linear gradient vector end point

Center of the ending circle in local coordinates

struct *lv\_grad\_dsc\_t* **linear**

*lv\_point\_t* **focal**

Center of the focal (starting) circle in local coordinates

***lv\_point\_t* **focal\_extent****

Point on the circle (can be the same as the center)

***lv\_point\_t* **end\_extent****

Point on the circle determining the radius of the gradient

struct *lv\_grad\_dsc\_t* **radial**

***lv\_point\_t* **center****

Conical gradient center point

**int16\_t **start\_angle****

Start angle 0..3600

**int16\_t **end\_angle****

End angle 0..3600

struct *lv\_grad\_dsc\_t* **conical**

union *lv\_grad\_dsc\_t* **params**

void \***state**

## 14.18.21 lv\_iter.h

### Typedef

typedef lv\_result\_t (\***lv\_iter\_next\_cb**)(void \*instance, void \*context, void \*elem)

typedef void (\***lv\_iter\_inspect\_cb**)(void \*elem)

### Functions

lv\_iter\_t \***lv\_iter\_create**(void \*instance, uint32\_t elem\_size, uint32\_t context\_size, *lv\_iter\_next\_cb* next\_cb)

Create an iterator based on an instance, and then the next element of the iterator can be obtained through *lv\_iter\_next*, In order to obtain the next operation in a unified and abstract way.

#### Parametri

- **instance** -- The instance to be iterated
- **elem\_size** -- The size of the element to be iterated in bytes
- **context\_size** -- The size of the context to be passed to the next\_cb in bytes
- **next\_cb** -- The callback function to get the next element

#### Ritorna

The iterator object

void \***lv\_iter\_get\_context**(const lv\_iter\_t \*iter)

Get the context of the iterator. You can use it to store some temporary variables associated with current iterator..

#### Parametri

**iter** -- *lv\_iter\_t* object create before

#### Ritorna

the iter context

**void lv\_iter\_destroy(lv\_iter\_t \*iter)**

Destroy the iterator object, and release the context. Other resources allocated by the user are not released. The user needs to release it by itself.

**Parametri**

**iter**-- `lv_iter_t` object create before

**lv\_result\_t lv\_iter\_next(lv\_iter\_t \*iter, void \*elem)**

Get the next element of the iterator.

**Parametri**

- **iter**-- `lv_iter_t` object create before
- **elem**-- The pointer to store the next element

**Ritorna**

LV\_RESULT\_OK: Get the next element successfully  
LV\_RESULT\_INVALID: The next element is invalid

**void lv\_iter\_make\_peekable(lv\_iter\_t \*iter, uint32\_t capacity)**

Make the iterator peekable, which means that the user can peek the next element without advancing the iterator.

**Parametri**

- **iter**-- `lv_iter_t` object create before
- **capacity**-- The capacity of the peek buffer

**lv\_result\_t lv\_iter\_peek(lv\_iter\_t \*iter, void \*elem)**

Peek the next element of the iterator without advancing the iterator.

**Parametri**

- **iter**-- `lv_iter_t` object create before
- **elem**-- The pointer to store the next element

**Ritorna**

LV\_RESULT\_OK: Peek the next element successfully  
LV\_RESULT\_INVALID: The next element is invalid

**lv\_result\_t lv\_iter\_peek\_advance(lv\_iter\_t \*iter)**

Only advance the iterator without getting the next element.

**Parametri**

**iter**-- `lv_iter_t` object create before

**Ritorna**

LV\_RESULT\_OK: Peek the next element successfully  
LV\_RESULT\_INVALID: The next element is invalid

**lv\_result\_t lv\_iter\_peek\_reset(lv\_iter\_t \*iter)**

Reset the peek cursor to the `next` cursor.

**Parametri**

**iter**-- `lv_iter_t` object create before

**Ritorna**

LV\_RESULT\_OK: Reset the peek buffer successfully  
LV\_RESULT\_INVALID: The peek buffer is invalid

**void lv\_iter\_inspect(lv\_iter\_t \*iter, *lv\_iter\_inspect\_cb* inspect\_cb)**

Inspect the element of the iterator. The callback function will be called for each element of the iterator.

**Parametri**

- **iter**-- `lv_iter_t` object create before
- **inspect\_cb**-- The callback function to inspect the element

## 14.18.22 lv\_ll.h

Handle linked lists. The nodes are dynamically allocated by the 'lv\_mem' module.

**Defines****LV\_LL\_READ**(list, i)**LV\_LL\_READ\_BACK**(list, i)**Typedef**typedef uint8\_t **lv\_ll\_node\_t**

Dummy type to make handling easier

**Functions**void **lv\_ll\_init**(*lv\_ll\_t* \*ll\_p, uint32\_t node\_size)

Initialize linked list

**Parametri**

- **ll\_p** -- pointer to *lv\_ll\_t* variable
- **node\_size** -- the size of 1 node in bytes

void \***lv\_ll\_ins\_head**(*lv\_ll\_t* \*ll\_p)

Add a new head to a linked list

**Parametri**

- **ll\_p** -- pointer to linked list

**Ritorna**

pointer to the new head

void \***lv\_ll\_ins\_prev**(*lv\_ll\_t* \*ll\_p, void \*n\_act)

Insert a new node in front of the n\_act node

**Parametri**

- **ll\_p** -- pointer to linked list
- **n\_act** -- pointer a node

**Ritorna**

pointer to the new node

void \***lv\_ll\_ins\_tail**(*lv\_ll\_t* \*ll\_p)

Add a new tail to a linked list

**Parametri**

- **ll\_p** -- pointer to linked list

**Ritorna**

pointer to the new tail

void **lv\_ll\_remove**(*lv\_ll\_t* \*ll\_p, void \*node\_p)

Remove the node 'node\_p' from 'll\_p' linked list. It does not free the memory of node.

**Parametri**

- **ll\_p** -- pointer to the linked list of 'node\_p'
- **node\_p** -- pointer to node in 'll\_p' linked list

void **lv\_ll\_clear\_custom**(*lv\_ll\_t* \*ll\_p, void (\*cleanup)(void\*))void **lv\_ll\_clear**(*lv\_ll\_t* \*ll\_p)

Remove and free all elements from a linked list. The list remain valid but become empty.

**Parametri**

- **ll\_p** -- pointer to linked list

void **lv\_ll\_chg\_list**(*lv\_ll\_t* \*ll\_ori\_p, *lv\_ll\_t* \*ll\_new\_p, void \*node, bool head)

Move a node to a new linked list

**Parametri**

- **ll\_ori\_p** -- pointer to the original (old) linked list
- **ll\_new\_p** -- pointer to the new linked list
- **node** -- pointer to a node

- **head** -- true: be the head in the new list false be the tail in the new list

`void *lv_ll_get_head(const lv_ll_t *ll_p)`

Return with head node of the linked list

**Parametri**

`ll_p` -- pointer to linked list

**Ritorna**

pointer to the head of 'll\_p'

`void *lv_ll_get_tail(const lv_ll_t *ll_p)`

Return with tail node of the linked list

**Parametri**

`ll_p` -- pointer to linked list

**Ritorna**

pointer to the tail of 'll\_p'

`void *lv_ll_get_next(const lv_ll_t *ll_p, const void *n_act)`

Return with the pointer of the next node after 'n\_act'

**Parametri**

- `ll_p` -- pointer to linked list
- `n_act` -- pointer a node

**Ritorna**

pointer to the next node

`void *lv_ll_get_prev(const lv_ll_t *ll_p, const void *n_act)`

Return with the pointer of the previous node after 'n\_act'

**Parametri**

- `ll_p` -- pointer to linked list
- `n_act` -- pointer a node

**Ritorna**

pointer to the previous node

`uint32_t lv_ll_get_len(const lv_ll_t *ll_p)`

Return the length of the linked list.

**Parametri**

`ll_p` -- pointer to linked list

**Ritorna**

length of the linked list

`void lv_ll_move_before(lv_ll_t *ll_p, void *n_act, void *n_after)`

Move a node before another node in the same linked list

**Parametri**

- `ll_p` -- pointer to a linked list
- `n_act` -- pointer to node to move
- `n_after` -- pointer to a node which should be after `n_act`

`bool lv_ll_is_empty(lv_ll_t *ll_p)`

Check if a linked list is empty

**Parametri**

`ll_p` -- pointer to a linked list

**Ritorna**

true: the linked list is empty; false: not empty

`struct lv_ll_t`

#include <lv\_ll.h> Description of a linked list

**Public Members****uint32\_t n\_size*****lv\_ll\_node\_t* \*head*****lv\_ll\_node\_t* \*tail****14.18.23 lv\_log.h****Defines****LV\_LOG\_LEVEL\_TRACE**

Log detailed information.

**LV\_LOG\_LEVEL\_INFO**

Log important events.

**LV\_LOG\_LEVEL\_WARN**

Log if something unwanted happened but didn't caused problem.

**LV\_LOG\_LEVEL\_ERROR**

Log only critical issues, when system may fail.

**LV\_LOG\_LEVEL\_USER**

Log only custom log messages added by the user.

**LV\_LOG\_LEVEL\_NONE**

Do not log anything.

**LV\_LOG\_LEVEL\_NUM**

Number of log levels

**LV\_LOG\_FILE****LV\_LOG\_LINE****LV\_LOG\_TRACE(...)****LV\_LOG\_INFO(...)****LV\_LOG\_WARN(...)****LV\_LOG\_ERROR(...)****LV\_LOG\_USER(...)****LV\_LOG(...)**

**Typedef**

```
typedef int8_t lv_log_level_t
```

```
typedef void (*lv_log_print_g_cb_t)(lv_log_level_t level, const char *buf)
```

Log print function. Receives a string buffer to print".

**Functions**

```
void lv_log_register_print_cb(lv_log_print_g_cb_t print_cb)
```

Register custom print/write function to call when a log is added. It can format its "File path", "Line number" and "Description" as required and send the formatted log message to a console or serial port.

**Parametri**

- **print\_cb** -- a function pointer to print a log

```
void lv_log(const char *format, ...)
```

Print a log message via `printf` if enabled with `LV_LOG_PRINTF` in `lv_conf.h` and/or a print callback if registered with `lv_log_register_print_cb`

**Parametri**

- **format** -- printf-like format string
- ... -- parameters for **format**

```
void lv_log_add(lv_log_level_t level, const char *file, int line, const char *func, const char *format, ...)
```

Add a log

**Parametri**

- **level** -- the level of log. (From `lv_log_level_t` enum)
- **file** -- name of the file when the log added
- **line** -- line number in the source code where the log added
- **func** -- name of the function when the log added
- **format** -- printf-like format string
- ... -- parameters for **format**

**14.18.24 lv\_lru.h****Typedef**

```
typedef void (*lv_lru_free_cb_t)(void *v)
```

```
typedef struct _lv_lru_item_t lv_lru_item_t
```

```
typedef struct _lv_lru_t lv_lru_t
```

**Enums**

```
enum lv_lru_res_t
```

*Values:*

enumerator **LV\_LRU\_OK**

enumerator **LV\_LRU\_MISSING\_CACHE**

enumerator **LV\_LRU\_MISSING\_KEY**

enumerator **LV\_LRU\_MISSING\_VALUE**

enumerator **LV\_LRU\_LOCK\_ERROR**

enumerator **LV\_LRU\_VALUE\_TOO\_LARGE**

## Functions

`lv_lru_t *lv_lru_create(size_t cache_size, size_t average_length, lv_lru_free_cb_t value_free,  
lv_lru_free_cb_t key_free)`

`void lv_lru_delete(lv_lru_t *cache)`

`lv_lru_res_t lv_lru_set(lv_lru_t *cache, const void *key, size_t key_length, void *value, size_t value_length)`

`lv_lru_res_t lv_lru_get(lv_lru_t *cache, const void *key, size_t key_size, void **value)`

`lv_lru_res_t lv_lru_remove(lv_lru_t *cache, const void *key, size_t key_size)`

`void lv_lru_remove_lru_item(lv_lru_t *cache)`

remove the least recently used item

*Todo:*

we can optimise this by finding the n lru items, where n = required\_space / average\_length

struct **\_lv\_lru\_t**

## Public Members

`lv_lru_item_t **items`

`uint64_t access_count`

`size_t free_memory`

`size_t total_memory`

`size_t average_item_length`

`size_t hash_table_size`

`uint32_t seed`

`lv_lru_free_cb_t value_free`

`lv_lru_free_cb_t key_free`

`lv_lru_item_t *free_items`

## 14.18.25 lv\_math.h

### Defines

**LV\_TRIGO\_SIN\_MAX**

**LV\_TRIGO\_SHIFT**

LV\_TRIGO\_SHIFT to normalize

**LV\_BEZIER\_VAL\_SHIFT**

$\log_2(LV\_BEZIER\_VAL\_MAX)$ : used to normalize up scaled values

**LV\_BEZIER\_VAL\_MAX**

Max time in Bezier functions (not [0..1] to use integers)

**LV\_BEZIER\_VAL\_FLOAT(f)**

Convert const float number cubic-bezier values to fix-point value

**LV\_ALIGN\_UP(x, align)**

Align up value x to align, align must be a power of two

**LV\_ROUND\_UP(x, round)**

Round up value x to round, round can be any integer number

**LV\_MIN(a, b)**

**LV\_MIN3(a, b, c)**

**LV\_MIN4(a, b, c, d)**

**LV\_MAX(a, b)**

**LV\_MAX3(a, b, c)**

**LV\_MAX4(a, b, c, d)**

**LV\_CLAMP(min, val, max)**

**LV\_ABS(x)**

**LV\_UDIV255(x)**

**LV\_IS\_SIGNED(t)**

**LV\_UMAX\_OF(t)**

**LV\_SMAX\_OF(t)**

**LV\_MAX\_OF(t)**

### Functions

int32\_t **lv\_trigo\_sin(int16\_t angle)**

Return with sinus of an angle

**Parametri**

**angle --**

**Ritorna**

sinus of 'angle'. sin(-90) = -32767, sin(90) = 32767

int32\_t **lv\_trigo\_cos(int16\_t angle)**

`int32_t lv_cubic_bezier(int32_t x, int32_t x1, int32_t y1, int32_t x2, int32_t y2)`

Calculate the y value of cubic-bezier(x1, y1, x2, y2) function as specified x.

**Parametri**

- **x** -- time in range of [0..LV\_BEZIER\_VAL\_MAX]
- **x1** -- x of control point 1 in range of [0..LV\_BEZIER\_VAL\_MAX]
- **y1** -- y of control point 1 in range of [0..LV\_BEZIER\_VAL\_MAX]
- **x2** -- x of control point 2 in range of [0..LV\_BEZIER\_VAL\_MAX]
- **y2** -- y of control point 2 in range of [0..LV\_BEZIER\_VAL\_MAX]

**Ritorna**

the value calculated

`int32_t lv_bezier3(int32_t t, int32_t u0, uint32_t u1, int32_t u2, int32_t u3)`

Calculate a value of a Cubic Bezier function.

**Parametri**

- **t** -- time in range of [0..LV\_BEZIER\_VAL\_MAX]
- **u0** -- must be 0
- **u1** -- control value 1 values in range of [0..LV\_BEZIER\_VAL\_MAX]
- **u2** -- control value 2 in range of [0..LV\_BEZIER\_VAL\_MAX]
- **u3** -- must be LV\_BEZIER\_VAL\_MAX

**Ritorna**

the value calculated from the given parameters in range of [0..LV\_BEZIER\_VAL\_MAX]

`uint16_t lv_atan2(int x, int y)`

Calculate the atan2 of a vector.

**Parametri**

- **x** --
- **y** --

**Ritorna**

the angle in degree calculated from the given parameters in range of [0..360]

`void lv_sqrt(uint32_t x, lv_sqrt_res_t *q, uint32_t mask)`

Get the square root of a number

**Parametri**

- **x** -- integer which square root should be calculated
- **q** -- store the result here. q->i: integer part, q->f: fractional part in 1/256 unit
- **mask** -- optional to skip some iterations if the magnitude of the root is known. Set to 0x8000 by default. If root < 16: mask = 0x80 If root < 256: mask = 0x800 Else: mask = 0x8000

`int32_t lv_sqrt32(uint32_t x)`

Alternative (fast, approximate) implementation for getting the square root of an integer.

**Parametri**

- **x** -- integer which square root should be calculated

`static inline int32_t lv_sqr(int32_t x)`

Calculate the square of an integer (input range is 0..32767).

**Parametri**

- **x** -- input

**Ritorna**

square

`int64_t lv_pow(int64_t base, int8_t exp)`

Calculate the integer exponents.

**Parametri**

- **base** --
- **exp** --

**Ritorna**

base raised to the power exponent

`int32_t lv_map(int32_t x, int32_t min_in, int32_t max_in, int32_t min_out, int32_t max_out)`

Get the mapped of a number given an input and output range

**Parametri**

- **x** -- integer which mapped value should be calculated
- **min\_in** -- min input range
- **max\_in** -- max input range
- **min\_out** -- max output range
- **max\_out** -- max output range

**Ritorna**

the mapped number

`void lv_rand_set_seed(uint32_t seed)`

Set the seed of the pseudo random number generator

**Parametri**

- **seed** -- a number to initialize the random generator

`uint32_t lv_rand(uint32_t min, uint32_t max)`

Get a pseudo random number in the given range

**Parametri**

- **min** -- the minimum value
- **max** -- the maximum value

**Ritorna**

return the random number. min <= return\_value <= max

struct `lv_sqrt_res_t`

### Public Members

`uint16_t i`

`uint16_t f`

## 14.18.26 lv\_matrix.h

### Functions

`void lv_matrix_identity(lv_matrix_t *matrix)`

Set matrix to identity matrix

**Parametri**

- **matrix** -- pointer to a matrix

`void lv_matrix_translate(lv_matrix_t *matrix, float tx, float ty)`

Translate the matrix to new position

**Parametri**

- **matrix** -- pointer to a matrix
- **tx** -- the amount of translate in x direction
- **ty** -- the amount of translate in y direction

`void lv_matrix_scale(lv_matrix_t *matrix, float scale_x, float scale_y)`

Change the scale factor of the matrix

**Parametri**

- **matrix** -- pointer to a matrix
- **scale\_x** -- the scale factor for the X direction
- **scale\_y** -- the scale factor for the Y direction

`void lv_matrix_rotate(lv_matrix_t *matrix, float degree)`

Rotate the matrix with origin

**Parametri**

- **matrix** -- pointer to a matrix
- **degree** -- angle to rotate

**void lv\_matrix\_skew(lv\_matrix\_t \*matrix, float skew\_x, float skew\_y)**

Change the skew factor of the matrix

**Parametri**

- **matrix** -- pointer to a matrix
- **skew\_x** -- the skew factor for x direction
- **skew\_y** -- the skew factor for y direction

**void lv\_matrix\_multiply(lv\_matrix\_t \*matrix, const lv\_matrix\_t \*mul)**

Multiply two matrix and store the result to the first one

**Parametri**

- **matrix** -- pointer to a matrix
- **matrix2** -- pointer to another matrix

**bool lv\_matrix\_inverse(lv\_matrix\_t \*matrix, const lv\_matrix\_t \*m)**

Invert the matrix

**Parametri**

- **matrix** -- pointer to a matrix
- **m** -- pointer to another matrix (optional)

**Ritorna**

true: the matrix is invertible, false: the matrix is singular and cannot be inverted

*lv\_point\_precise\_t* **lv\_matrix\_transform\_precise\_point**(const lv\_matrix\_t \*matrix, const *lv\_point\_precise\_t* \*point)

Transform a point by a matrix

**Parametri**

- **matrix** -- pointer to a matrix
- **point** -- pointer to a point

**Ritorna**

the transformed point

*lv\_area\_t* **lv\_matrix\_transform\_area**(const lv\_matrix\_t \*matrix, const *lv\_area\_t* \*area)

Transform an area by a matrix

**Parametri**

- **matrix** -- pointer to a matrix
- **area** -- pointer to an area

**Ritorna**

the transformed area

**bool lv\_matrix\_is\_identity(const lv\_matrix\_t \*matrix)**

Check if the matrix is identity

**Parametri**

**matrix** -- pointer to a matrix

**Ritorna**

true: the matrix is identity , false: the matrix is not identity

**bool lv\_matrix\_is\_identity\_or\_translation(const lv\_matrix\_t \*matrix)**

Check if the matrix is identity or translation matrix

**Parametri**

**matrix** -- pointer to a matrix

**Ritorna**

true: the matrix is identity or translation matrix, false: the matrix is not identity or translation matrix

struct **\_lv\_matrix\_t**

## Public Members

float **m**[3][3]

### 14.18.27 lv\_palette.h

#### Enums

enum **lv\_palette\_t**

*Values:*

enumerator **LV\_PALETTE\_RED**

enumerator **LV\_PALETTE\_PINK**

enumerator **LV\_PALETTE\_PURPLE**

enumerator **LV\_PALETTE\_DEEP\_PURPLE**

enumerator **LV\_PALETTE\_INDIGO**

enumerator **LV\_PALETTE\_BLUE**

enumerator **LV\_PALETTE\_LIGHT\_BLUE**

enumerator **LV\_PALETTE\_CYAN**

enumerator **LV\_PALETTE\_TEAL**

enumerator **LV\_PALETTE\_GREEN**

enumerator **LV\_PALETTE\_LIGHT\_GREEN**

enumerator **LV\_PALETTE\_LIME**

enumerator **LV\_PALETTE\_YELLOW**

enumerator **LV\_PALETTE\_AMBER**

enumerator **LV\_PALETTE\_ORANGE**

enumerator **LV\_PALETTE\_DEEP\_ORANGE**

enumerator **LV\_PALETTE\_BROWN**

enumerator **LV\_PALETTE\_BLUE\_GREY**

enumerator **LV\_PALETTE\_GREY**

enumerator **LV\_PALETTE\_LAST**

enumerator **LV\_PALETTE\_NONE**

### Functions

*lv\_color\_t* **lv\_palette\_main**(*lv\_palette\_t* p)

*lv\_color\_t* **lv\_palette\_lighten**(*lv\_palette\_t* p, *uint8\_t* lvl)

*lv\_color\_t* **lv\_palette\_darken**(*lv\_palette\_t* p, *uint8\_t* lvl)

## 14.18.28 lv\_profiler.h

### Defines

**LV\_PROFILER\_BEGIN**

**LV\_PROFILER\_END**

**LV\_PROFILER\_BEGIN\_TAG**(tag)

**LV\_PROFILER\_END\_TAG**(tag)

**LV\_PROFILER\_LAYOUT\_BEGIN**

**LV\_PROFILER\_LAYOUT\_END**

**LV\_PROFILER\_LAYOUT\_BEGIN\_TAG**(tag)

**LV\_PROFILER\_LAYOUT\_END\_TAG**(tag)

**LV\_PROFILER\_STYLE\_BEGIN**

**LV\_PROFILER\_STYLE\_END**

**LV\_PROFILER\_STYLE\_BEGIN\_TAG**(tag)

**LV\_PROFILER\_STYLE\_END\_TAG**(tag)

**LV\_PROFILER\_DRAW\_BEGIN**

**LV\_PROFILER\_DRAW\_END**

**LV\_PROFILER\_DRAW\_BEGIN\_TAG**(tag)

**LV\_PROFILER\_DRAW\_END\_TAG**(tag)

**LV\_PROFILER\_DECODER\_BEGIN**

**LV\_PROFILER\_DECODER\_END**

`LV_PROFILER_DECODER_BEGIN_TAG(tag)`

`LV_PROFILER_DECODER_END_TAG(tag)`

`LV_PROFILER_REFR_BEGIN`

`LV_PROFILER_REFR_END`

`LV_PROFILER_REFR_BEGIN_TAG(tag)`

`LV_PROFILER_REFR_END_TAG(tag)`

`LV_PROFILER_INDEV_BEGIN`

`LV_PROFILER_INDEV_END`

`LV_PROFILER_INDEV_BEGIN_TAG(tag)`

`LV_PROFILER_INDEV_END_TAG(tag)`

`LV_PROFILER_FONT_BEGIN`

`LV_PROFILER_FONT_END`

`LV_PROFILER_FONT_BEGIN_TAG(tag)`

`LV_PROFILER_FONT_END_TAG(tag)`

`LV_PROFILER_CACHE_BEGIN`

`LV_PROFILER_CACHE_END`

`LV_PROFILER_CACHE_BEGIN_TAG(tag)`

`LV_PROFILER_CACHE_END_TAG(tag)`

`LV_PROFILER_FS_BEGIN`

`LV_PROFILER_FS_END`

`LV_PROFILER_FS_BEGIN_TAG(tag)`

`LV_PROFILER_FS_END_TAG(tag)`

`LV_PROFILER_TIMER_BEGIN`

`LV_PROFILER_TIMER_END`

`LV_PROFILER_TIMER_BEGIN_TAG(tag)`

`LV_PROFILER_TIMER_END_TAG(tag)`

`LV_PROFILER_EVENT_BEGIN`

**LV\_PROFILER\_EVENT\_END****LV\_PROFILER\_EVENT\_BEGIN\_TAG**(tag)**LV\_PROFILER\_EVENT\_END\_TAG**(tag)

### **14.18.29 lv\_profiler\_builtin.h**

### **14.18.30 lv\_profiler\_builtin\_private.h**

### **14.18.31 lv\_rb.h**

#### **Typedef**

**typedef int8\_t lv\_rb\_compare\_res\_t****typedef lv\_rb\_compare\_res\_t (\*lv\_rb\_compare\_t)(const void \*a, const void \*b)**

#### **Enums**

**enum lv\_rb\_color\_t***Values:*enumerator **LV\_RB\_COLOR\_RED**enumerator **LV\_RB\_COLOR\_BLACK**

#### **Functions**

**bool lv\_rb\_init(lv\_rb\_t \*tree, lv\_rb\_compare\_t compare, size\_t node\_size)****lv\_rb\_node\_t \*lv\_rb\_insert(lv\_rb\_t \*tree, void \*key)****lv\_rb\_node\_t \*lv\_rb\_find(lv\_rb\_t \*tree, const void \*key)****void \*lv\_rb\_remove\_node(lv\_rb\_t \*tree, lv\_rb\_node\_t \*node)****void \*lv\_rb\_remove(lv\_rb\_t \*tree, const void \*key)****bool lv\_rb\_drop\_node(lv\_rb\_t \*tree, lv\_rb\_node\_t \*node)****bool lv\_rb\_drop(lv\_rb\_t \*tree, const void \*key)****lv\_rb\_node\_t \*lv\_rb\_minimum(lv\_rb\_t \*node)****lv\_rb\_node\_t \*lv\_rb\_maximum(lv\_rb\_t \*node)****lv\_rb\_node\_t \*lv\_rb\_minimum\_from(lv\_rb\_node\_t \*node)****lv\_rb\_node\_t \*lv\_rb\_maximum\_from(lv\_rb\_node\_t \*node)****void lv\_rb\_destroy(lv\_rb\_t \*tree)**

### **14.18.32 lv\_rb\_private.h**

**struct \_lv\_rb\_node\_t**

**Public Members**

```
struct _lv_rb_node_t *parent
struct _lv_rb_node_t *left
struct _lv_rb_node_t *right
lv_rb_color_t color
void *data
struct _lv_rb_t
```

**Public Members**

```
lv_rb_node_t *root
lv_rb_compare_t compare
size_t size
```

**14.18.33 lv\_style.h****Defines****LV\_STYLE\_SENTINEL\_VALUE****LV\_STYLE\_PROP\_FLAG\_NONE**

No special behavior

**LV\_STYLE\_PROP\_FLAG\_INHERITABLE**

Inherited

**LV\_STYLE\_PROP\_FLAG\_EXT\_DRAW\_UPDATE**

Requires ext. draw size update when changed

**LV\_STYLE\_PROP\_FLAG\_LAYOUT\_UPDATE**

Requires layout update when changed

**LV\_STYLE\_PROP\_FLAG\_PARENT\_LAYOUT\_UPDATE**

Requires layout update on parent when changed

**LV\_STYLE\_PROP\_FLAG\_LAYER\_UPDATE**

Affects layer handling

**LV\_STYLE\_PROP\_FLAG\_TRANSFORM**

Affects the object's transformation

**LV\_STYLE\_PROP\_FLAG\_ALL**

Indicating all flags

**LV\_SCALE\_NONE**

Value for not zooming the image

**LV\_STYLE\_CONST\_INIT**(var\_name, prop\_array)**LV\_STYLE\_CONST\_PROPS\_END****LV\_GRAD\_LEFT****LV\_GRAD\_RIGHT****LV\_GRAD\_TOP****LV\_GRAD\_BOTTOM****LV\_GRAD\_CENTER****LV\_ASSERT\_STYLE**(style\_p)**Enums****enum lv\_blend\_mode\_t**

Possible options for blending opaque drawings

*Values:*

**enumerator LV\_BLEND\_MODE\_NORMAL**

Simply mix according to the opacity value

**enumerator LV\_BLEND\_MODE\_ADDITIVE**

Add the respective color channels

**enumerator LV\_BLEND\_MODE\_SUBTRACTIVE**

Subtract the foreground from the background

**enumerator LV\_BLEND\_MODE\_MULTIPLY**

Multiply the foreground and background

**enumerator LV\_BLEND\_MODE\_DIFFERENCE**

Absolute difference between foreground and background

**enum lv\_text\_decor\_t**

Some options to apply decorations on texts. 'OR'ed values can be used.

*Values:*

**enumerator LV\_TEXT\_DECOR\_NONE**

enumerator **LV\_TEXT\_DECOR\_UNDERLINE**

enumerator **LV\_TEXT\_DECOR\_STRIKETHROUGH**

enum **lv\_border\_side\_t**

Selects on which sides border should be drawn 'OR'ed values can be used.

*Values:*

enumerator **LV\_BORDER\_SIDE\_NONE**

enumerator **LV\_BORDER\_SIDE\_BOTTOM**

enumerator **LV\_BORDER\_SIDE\_TOP**

enumerator **LV\_BORDER\_SIDE\_LEFT**

enumerator **LV\_BORDER\_SIDE\_RIGHT**

enumerator **LV\_BORDER\_SIDE\_FULL**

enumerator **LV\_BORDER\_SIDE\_INTERNAL**

FOR matrix-like objects (e.g. Button matrix)

enum **\_lv\_style\_id\_t**

Enumeration of all built in style properties

Props are split into groups of 16. When adding a new prop to a group, ensure it does not overflow into the next one.

*Values:*

enumerator **LV\_STYLE\_PROP\_INV**

enumerator **LV\_STYLE\_WIDTH**

enumerator **LV\_STYLE\_HEIGHT**

enumerator **LV\_STYLE\_LENGTH**

enumerator **LV\_STYLE\_MIN\_WIDTH**

enumerator **LV\_STYLE\_MAX\_WIDTH**

enumerator **LV\_STYLE\_MIN\_HEIGHT**

enumerator **LV\_STYLE\_MAX\_HEIGHT**

enumerator **LV\_STYLE\_X**

enumerator **LV\_STYLE\_Y**

enumerator **LV\_STYLE\_ALIGN**

enumerator **LV\_STYLE\_RADIUS**

enumerator **LV\_STYLE\_RADIAL\_OFFSET**

enumerator **LV\_STYLE\_PAD\_RADIAL**

enumerator **LV\_STYLE\_PAD\_TOP**

enumerator **LV\_STYLE\_PAD\_BOTTOM**

enumerator **LV\_STYLE\_PAD\_LEFT**

enumerator **LV\_STYLE\_PAD\_RIGHT**

enumerator **LV\_STYLE\_PAD\_ROW**

enumerator **LV\_STYLE\_PAD\_COLUMN**

enumerator **LV\_STYLE\_LAYOUT**

enumerator **LV\_STYLE\_MARGIN\_TOP**

enumerator **LV\_STYLE\_MARGIN\_BOTTOM**

enumerator **LV\_STYLE\_MARGIN\_LEFT**

enumerator **LV\_STYLE\_MARGIN\_RIGHT**

enumerator **LV\_STYLE\_BG\_COLOR**

enumerator **LV\_STYLE\_BG\_OPA**

enumerator **LV\_STYLE\_BG\_GRAD\_DIR**

enumerator **LV\_STYLE\_BG\_MAIN\_STOP**

enumerator **LV\_STYLE\_BG\_GRAD\_STOP**

enumerator **LV\_STYLE\_BG\_GRAD\_COLOR**

enumerator **LV\_STYLE\_BG\_MAIN\_OPA**

enumerator **LV\_STYLE\_BG\_GRAD\_OPA**

enumerator **LV\_STYLE\_BG\_GRAD**

enumerator **LV\_STYLE\_BASE\_DIR**

enumerator **LV\_STYLE\_BG\_IMAGE\_SRC**

enumerator **LV\_STYLE\_BG\_IMAGE\_OPA**

enumerator **LV\_STYLE\_BG\_IMAGE\_RECOLOR**

enumerator **LV\_STYLE\_BG\_IMAGE\_RECOLOR\_OPA**

enumerator **LV\_STYLE\_BG\_IMAGE\_TILED**

enumerator **LV\_STYLE\_CLIP\_CORNER**

enumerator **LV\_STYLE\_BORDER\_WIDTH**

enumerator **LV\_STYLE\_BORDER\_COLOR**

enumerator **LV\_STYLE\_BORDER\_OPA**

enumerator **LV\_STYLE\_BORDER\_SIDE**

enumerator **LV\_STYLE\_BORDER\_POST**

enumerator **LV\_STYLE\_OUTLINE\_WIDTH**

enumerator **LV\_STYLE\_OUTLINE\_COLOR**

enumerator **LV\_STYLE\_OUTLINE\_OPA**

enumerator **LV\_STYLE\_OUTLINE\_PAD**

enumerator **LV\_STYLE\_SHADOW\_WIDTH**

enumerator **LV\_STYLE\_SHADOW\_COLOR**

enumerator **LV\_STYLE\_SHADOW\_OPA**

enumerator **LV\_STYLE\_SHADOW\_OFFSET\_X**

enumerator **LV\_STYLE\_SHADOW\_OFFSET\_Y**

enumerator **LV\_STYLE\_SHADOW\_SPREAD**

enumerator **LV\_STYLE\_IMAGE\_OPA**

enumerator **LV\_STYLE\_IMAGE\_RECOLOR**

enumerator **LV\_STYLE\_IMAGE\_RECOLOR\_OPA**

enumerator **LV\_STYLE\_LINE\_WIDTH**

enumerator **LV\_STYLE\_LINE\_DASH\_WIDTH**

enumerator **LV\_STYLE\_LINE\_DASH\_GAP**

enumerator **LV\_STYLE\_LINE\_ROUNDED**

enumerator **LV\_STYLE\_LINE\_COLOR**

enumerator **LV\_STYLE\_LINE\_OPA**

enumerator **LV\_STYLE\_ARC\_WIDTH**

enumerator **LV\_STYLE\_ARC\_ROUNDED**

enumerator **LV\_STYLE\_ARC\_COLOR**

enumerator **LV\_STYLE\_ARC\_OPA**

enumerator **LV\_STYLE\_ARC\_IMAGE\_SRC**

enumerator **LV\_STYLE\_TEXT\_COLOR**

enumerator **LV\_STYLE\_TEXT\_OPA**

enumerator **LV\_STYLE\_TEXT\_FONT**

enumerator **LV\_STYLE\_TEXT\_LETTER\_SPACE**

enumerator **LV\_STYLE\_TEXT\_LINE\_SPACE**

enumerator **LV\_STYLE\_TEXT\_DECOR**

enumerator **LV\_STYLE\_TEXT\_ALIGN**

enumerator **LV\_STYLE\_TEXT\_OUTLINE\_STROKE\_WIDTH**

enumerator **LV\_STYLE\_TEXT\_OUTLINE\_STROKE\_OPA**

enumerator **LV\_STYLE\_TEXT\_OUTLINE\_STROKE\_COLOR**

enumerator **LV\_STYLE\_OPA**

enumerator **LV\_STYLE\_OPA\_LAYERED**

enumerator **LV\_STYLE\_COLOR\_FILTER\_DSC**

enumerator **LV\_STYLE\_COLOR\_FILTER\_OPA**

enumerator **LV\_STYLE\_ANIM**

enumerator **LV\_STYLE\_ANIM\_DURATION**

enumerator **LV\_STYLE\_TRANSITION**

enumerator **LV\_STYLE\_BLEND\_MODE**

enumerator **LV\_STYLE\_TRANSFORM\_WIDTH**

enumerator **LV\_STYLE\_TRANSFORM\_HEIGHT**

enumerator **LV\_STYLE\_TRANSLATE\_X**

enumerator **LV\_STYLE\_TRANSLATE\_Y**

enumerator **LV\_STYLE\_TRANSFORM\_SCALE\_X**

enumerator **LV\_STYLE\_TRANSFORM\_SCALE\_Y**

enumerator **LV\_STYLE\_TRANSFORM\_ROTATION**

enumerator **LV\_STYLE\_TRANSFORM\_PIVOT\_X**

enumerator **LV\_STYLE\_TRANSFORM\_PIVOT\_Y**

enumerator **LV\_STYLE\_TRANSFORM\_SKW\_X**

enumerator **LV\_STYLE\_TRANSFORM\_SKW\_Y**

enumerator **LV\_STYLE\_BITMAP\_MASK\_SRC**

enumerator **LV\_STYLE\_ROTARY\_SENSITIVITY**

enumerator **LV\_STYLE\_TRANSLATE\_RADIAL**

enumerator **LV\_STYLE\_RECOLOR**

enumerator **LV\_STYLE\_RECOLOR\_OPA**

enumerator **LV\_STYLE\_FLEX\_FLOW**

enumerator **LV\_STYLE\_FLEX\_MAIN\_PLACE**

enumerator **LV\_STYLE\_FLEX\_CROSS\_PLACE**

enumerator **LV\_STYLE\_FLEX\_TRACK\_PLACE**

enumerator **LV\_STYLE\_FLEX\_GROW**

enumerator **LV\_STYLE\_GRID\_COLUMN\_ALIGN**

enumerator **LV\_STYLE\_GRID\_ROW\_ALIGN**

enumerator **LV\_STYLE\_GRID\_ROW\_DSC\_ARRAY**

enumerator **LV\_STYLE\_GRID\_COLUMN\_DSC\_ARRAY**

enumerator **LV\_STYLE\_GRID\_CELL\_COLUMN\_POS**

enumerator **LV\_STYLE\_GRID\_CELL\_COLUMN\_SPAN**

enumerator **LV\_STYLE\_GRID\_CELL\_X\_ALIGN**

enumerator **LV\_STYLE\_GRID\_CELL\_ROW\_POS**

enumerator **LV\_STYLE\_GRID\_CELL\_ROW\_SPAN**

enumerator **LV\_STYLE\_GRID\_CELL\_Y\_ALIGN**

enumerator **LV\_STYLE\_LAST\_BUILT\_IN\_PROP**

enumerator **LV\_STYLE\_NUM\_BUILT\_IN\_PROPS**

enumerator **LV\_STYLE\_PROP\_ANY**

enumerator **LV\_STYLE\_PROP\_CONST**

enum **lv\_style\_res\_t**

*Values:*

enumerator **LV\_STYLE\_RES\_NOT\_FOUND**

enumerator **LV\_STYLE\_RES\_FOUND**

## Functions

void **lv\_style\_init**(*lv\_style\_t* \*style)

Initialize a style

### Nota

Do not call **lv\_style\_init** on styles that already have some properties because this function won't free the used memory, just sets a default state for the style. In other words be sure to initialize styles only once!

#### Parametri

**style** -- pointer to a style to initialize

void **lv\_style\_reset**(*lv\_style\_t* \*style)

Clear all properties from a style and free all allocated memories.

#### Parametri

**style** -- pointer to a style

void **lv\_style\_copy**(*lv\_style\_t* \*dst, const *lv\_style\_t* \*src)

Copy all properties of a style to an other. It has the same affect calling the same **lv\_set\_style\_...** functions on both styles. It means new memory will be allocated to store the properties in the destination style. After the copy the destination style is fully independent of the source and source can removed without affecting the destination style.

#### Parametri

- **dst** -- the destination to copy into (can not be a constant style)
- **src** -- the source style to copy from.

static inline bool **lv\_style\_is\_const**(const *lv\_style\_t* \*style)

Check if a style is constant

#### Parametri

**style** -- pointer to a style

#### Ritorna

true: the style is constant

*lv\_style\_prop\_t* **lv\_style\_register\_prop**(uint8\_t flag)

Register a new style property for custom usage

Example:

```
lv_style_prop_t MY_PROP;
static inline void lv_style_set_my_prop(lv_style_t * style, lv_color_t value) {
    lv_style_value_t v = {.color = value}; lv_style_set_prop(style, MY_PROP, v);
}

...
MY_PROP = lv_style_register_prop();
...
lv_style_set_my_prop(&style1, lv_palette_main(LV_PALETTE_RED));
```

#### Ritorna

a new property ID, or **LV\_STYLE\_PROP\_INV** if there are no more available.

*lv\_style\_prop\_t* **lv\_style\_get\_num\_custom\_props**(void)

Get the number of custom properties that have been registered thus far.

`bool lv_style_remove_prop(lv_style_t *style, lv_style_prop_t prop)`

Remove a property from a style

**Parametri**

- **style** -- pointer to a style
- **prop** -- a style property ORed with a state.

**Ritorna**

true: the property was found and removed; false: the property wasn't found

`void lv_style_set_prop(lv_style_t *style, lv_style_prop_t prop, lv_style_value_t value)`

Set the value of property in a style. This function shouldn't be used directly by the user. Instead use `lv_style_set_<prop_name>()`. E.g. `lv_style_set_bg_color()`

**Parametri**

- **style** -- pointer to style
- **prop** -- the ID of a property (e.g. `LV_STYLE_BG_COLOR`)
- **value** -- `lv_style_value_t` variable in which a field is set according to the type of `prop`

`lv_style_res_t lv_style_get_prop(const lv_style_t *style, lv_style_prop_t prop, lv_style_value_t *value)`

Get the value of a property

**Nota**

For performance reasons there are no sanity check on `style`

**Parametri**

- **style** -- pointer to a style
- **prop** -- the ID of a property
- **value** -- pointer to a `lv_style_value_t` variable to store the value

**Ritorna**

`LV_RESULT_INVALID`: the property wasn't found in the style (`value` is unchanged)

`LV_RESULT_OK`: the property was fond, and `value` is set accordingly

`void lv_style_transition_dsc_init(lv_style_transition_dsc_t *tr, const lv_style_prop_t props[], lv_anim_path_cb_t path_cb, uint32_t time, uint32_t delay, void *user_data)`

Initialize a transition descriptor.

Example:

```
const static lv_style_prop_t trans_props[] = { LV_STYLE_BG_OPA, LV_STYLE_BG_COLOR, 0 };
static lv_style_transition_dsc_t transl;
lv_style_transition_dsc_init(&transl, trans_props, NULL, 300, 0, NULL);
```

**Parametri**

- **tr** -- pointer to a transition descriptor to initialize
- **props** -- an array with the properties to transition. The last element must be zero.
- **path\_cb** -- an animation path (ease) callback. If `NULL` liner path will be used.
- **time** -- duration of the transition in [ms]
- **delay** -- delay before the transition in [ms]
- **user\_data** -- any custom data that will be saved in the transition animation and will be available when `path_cb` is called

`lv_style_value_t lv_style_prop_get_default(lv_style_prop_t prop)`

Get the default value of a property

**Parametri**

- **prop** -- the ID of a property

**Ritorna**

the default value

```
static inline lv_style_res_t lv_style_get_prop_inlined(const lv_style_t *style, lv_style_prop_t prop,
lv_style_value_t *value)
```

Get the value of a property

**Nota**

For performance reasons there are no sanity check on **style**

**Nota**

This function is the same as *lv\_style\_get\_prop* but inlined. Use it only on performance critical places

**Parametri**

- **style** -- pointer to a style
- **prop** -- the ID of a property
- **value** -- pointer to a *lv\_style\_value\_t* variable to store the value

**Ritorna**

LV\_RESULT\_INVALID: the property wasn't found in the style (**value** is unchanged)

LV\_RESULT\_OK: the property was found, and **value** is set accordingly

```
bool lv_style_is_empty(const lv_style_t *style)
```

Checks if a style is empty (has no properties)

**Parametri**

- **style** -- pointer to a style

**Ritorna**

true if the style is empty

```
static inline uint32_t lv_style_get_prop_group(lv_style_prop_t prop)
```

Tell the group of a property. If the a property from a group is set in a style the (1 << group) bit of *style->has\_group* is set. It allows early skipping the style if the property is not exists in the style at all.

**Parametri**

- **prop** -- a style property

**Ritorna**

the group [0..30] 30 means all the custom properties with index > 120

```
uint8_t lv_style_prop_lookup_flags(lv_style_prop_t prop)
```

Get the flags of a built-in or custom property.

**Parametri**

- **prop** -- a style property

**Ritorna**

the flags of the property

```
static inline void lv_style_set_size(lv_style_t *style, int32_t width, int32_t height)
```

```
static inline void lv_style_set_pad_all(lv_style_t *style, int32_t value)
```

```
static inline void lv_style_set_pad_hor(lv_style_t *style, int32_t value)
```

```
static inline void lv_style_set_pad_ver(lv_style_t *style, int32_t value)
```

```
static inline void lv_style_set_pad_gap(lv_style_t *style, int32_t value)
```

```
static inline void lv_style_set_margin_hor(lv_style_t *style, int32_t value)
```

```
static inline void lv_style_set_margin_ver(lv_style_t *style, int32_t value)
```

```
static inline void lv_style_set_margin_all(lv_style_t *style, int32_t value)
```

static inline void **lv\_style\_set\_transform\_scale**(*lv\_style\_t* \*style, int32\_t value)

static inline bool **lv\_style\_prop\_has\_flag**(lv\_style\_prop\_t prop, uint8\_t flag)

Check if the style property has a specified behavioral flag.

Do not pass multiple flags to this function as backwards-compatibility is not guaranteed for that.

#### Parametri

- **prop** -- Property ID
- **flag** -- Flag

#### Ritorna

true if the flag is set for this property

## Variables

const lv\_style\_prop\_t **lv\_style\_const\_prop\_id\_inv**

union **lv\_style\_value\_t**

#include <lv\_style.h> A common type to handle all the property types in the same way.

## Public Members

int32\_t **num**

Number integer number (opacity, enums, booleans or "normal" numbers)

const void \***ptr**

Constant pointers (font, cone text, etc)

*lv\_color\_t* **color**

Colors

struct **lv\_style\_transition\_dsc\_t**

#include <lv\_style.h> Descriptor for style transitions

## Public Members

const lv\_style\_prop\_t \***props**

An array with the properties to animate.

void \***user\_data**

A custom user data that will be passed to the animation's user\_data

*lv\_anim\_path\_cb\_t* **path\_xcb**

A path for the animation.

uint32\_t **time**

Duration of the transition in [ms]

uint32\_t **delay**

Delay before the transition in [ms]

struct **lv\_style\_const\_prop\_t**

#include <lv\_style.h> Descriptor of a constant style property.

**Public Members**

```
lv_style_prop_t prop
```

```
lv_style_value_t value
```

```
struct lv_style_t
```

#include <lv\_style.h> Descriptor of a style (a collection of properties and values).

**Public Members**

```
uint32_t sentinel
```

```
void *values_and_props
```

```
uint32_t has_group
```

```
uint8_t prop_cnt
```

255 means it's a constant style

**14.18.34 lv\_style\_gen.h****Defines**

```
LV_STYLE_CONST_WIDTH(val)
```

```
LV_STYLE_CONST_MIN_WIDTH(val)
```

```
LV_STYLE_CONST_MAX_WIDTH(val)
```

```
LV_STYLE_CONST_HEIGHT(val)
```

```
LV_STYLE_CONST_MIN_HEIGHT(val)
```

```
LV_STYLE_CONST_MAX_HEIGHT(val)
```

```
LV_STYLE_CONST_LENGTH(val)
```

```
LV_STYLE_CONST_X(val)
```

```
LV_STYLE_CONST_Y(val)
```

```
LV_STYLE_CONST_ALIGN(val)
```

```
LV_STYLE_CONST_TRANSFORM_WIDTH(val)
```

```
LV_STYLE_CONST_TRANSFORM_HEIGHT(val)
```

```
LV_STYLE_CONST_TRANSLATE_X(val)
```

```
LV_STYLE_CONST_TRANSLATE_Y(val)
```

```
LV_STYLE_CONST_TRANSLATE_RADIAL(val)
```

```
LV_STYLE_CONST_TRANSFORM_SCALE_X(val)
```

```
LV_STYLE_CONST_TRANSFORM_SCALE_Y(val)
```

**LV\_STYLE\_CONST\_TRANSFORM\_ROTATION**(val)  
**LV\_STYLE\_CONST\_TRANSFORM\_PIVOT\_X**(val)  
**LV\_STYLE\_CONST\_TRANSFORM\_PIVOT\_Y**(val)  
**LV\_STYLE\_CONST\_TRANSFORM\_SKEW\_X**(val)  
**LV\_STYLE\_CONST\_TRANSFORM\_SKEW\_Y**(val)  
**LV\_STYLE\_CONST\_PAD\_TOP**(val)  
**LV\_STYLE\_CONST\_PAD\_BOTTOM**(val)  
**LV\_STYLE\_CONST\_PAD\_LEFT**(val)  
**LV\_STYLE\_CONST\_PAD\_RIGHT**(val)  
**LV\_STYLE\_CONST\_PAD\_ROW**(val)  
**LV\_STYLE\_CONST\_PAD\_COLUMN**(val)  
**LV\_STYLE\_CONST\_PAD\_RADIAL**(val)  
**LV\_STYLE\_CONST\_MARGIN\_TOP**(val)  
**LV\_STYLE\_CONST\_MARGIN\_BOTTOM**(val)  
**LV\_STYLE\_CONST\_MARGIN\_LEFT**(val)  
**LV\_STYLE\_CONST\_MARGIN\_RIGHT**(val)  
**LV\_STYLE\_CONST\_BG\_COLOR**(val)  
**LV\_STYLE\_CONST\_BG\_OPA**(val)  
**LV\_STYLE\_CONST\_BG\_GRAD\_COLOR**(val)  
**LV\_STYLE\_CONST\_BG\_GRAD\_DIR**(val)  
**LV\_STYLE\_CONST\_BG\_MAIN\_STOP**(val)  
**LV\_STYLE\_CONST\_BG\_GRAD\_STOP**(val)  
**LV\_STYLE\_CONST\_BG\_MAIN\_OPA**(val)  
**LV\_STYLE\_CONST\_BG\_GRAD\_OPA**(val)  
**LV\_STYLE\_CONST\_BG\_GRAD**(val)  
**LV\_STYLE\_CONST\_BG\_IMAGE\_SRC**(val)  
**LV\_STYLE\_CONST\_BG\_IMAGE\_OPA**(val)  
**LV\_STYLE\_CONST\_BG\_IMAGE\_RECOLOR**(val)  
**LV\_STYLE\_CONST\_BG\_IMAGE\_RECOLOR\_OPA**(val)  
**LV\_STYLE\_CONST\_BG\_IMAGE\_TILED**(val)  
**LV\_STYLE\_CONST\_BORDER\_COLOR**(val)  
**LV\_STYLE\_CONST\_BORDER\_OPA**(val)  
**LV\_STYLE\_CONST\_BORDER\_WIDTH**(val)

**LV\_STYLE\_CONST\_BORDER\_SIDE**(val)  
**LV\_STYLE\_CONST\_BORDER\_POST**(val)  
**LV\_STYLE\_CONST\_OUTLINE\_WIDTH**(val)  
**LV\_STYLE\_CONST\_OUTLINE\_COLOR**(val)  
**LV\_STYLE\_CONST\_OUTLINE\_OPA**(val)  
**LV\_STYLE\_CONST\_OUTLINE\_PAD**(val)  
**LV\_STYLE\_CONST\_SHADOW\_WIDTH**(val)  
**LV\_STYLE\_CONST\_SHADOW\_OFFSET\_X**(val)  
**LV\_STYLE\_CONST\_SHADOW\_OFFSET\_Y**(val)  
**LV\_STYLE\_CONST\_SHADOW\_SPREAD**(val)  
**LV\_STYLE\_CONST\_SHADOW\_COLOR**(val)  
**LV\_STYLE\_CONST\_SHADOW\_OPA**(val)  
**LV\_STYLE\_CONST\_IMAGE\_OPA**(val)  
**LV\_STYLE\_CONST\_IMAGE\_RECOLOR**(val)  
**LV\_STYLE\_CONST\_IMAGE\_RECOLOR\_OPA**(val)  
**LV\_STYLE\_CONST\_LINE\_WIDTH**(val)  
**LV\_STYLE\_CONST\_LINE\_DASH\_WIDTH**(val)  
**LV\_STYLE\_CONST\_LINE\_DASH\_GAP**(val)  
**LV\_STYLE\_CONST\_LINE\_ROUNDED**(val)  
**LV\_STYLE\_CONST\_LINE\_COLOR**(val)  
**LV\_STYLE\_CONST\_LINE\_OPA**(val)  
**LV\_STYLE\_CONST\_ARC\_WIDTH**(val)  
**LV\_STYLE\_CONST\_ARC\_ROUNDED**(val)  
**LV\_STYLE\_CONST\_ARC\_COLOR**(val)  
**LV\_STYLE\_CONST\_ARC\_OPA**(val)  
**LV\_STYLE\_CONST\_ARC\_IMAGE\_SRC**(val)  
**LV\_STYLE\_CONST\_TEXT\_COLOR**(val)  
**LV\_STYLE\_CONST\_TEXT\_OPA**(val)  
**LV\_STYLE\_CONST\_TEXT\_FONT**(val)  
**LV\_STYLE\_CONST\_TEXT\_LETTER\_SPACE**(val)  
**LV\_STYLE\_CONST\_TEXT\_LINE\_SPACE**(val)  
**LV\_STYLE\_CONST\_TEXT\_DECOR**(val)  
**LV\_STYLE\_CONST\_TEXT\_ALIGN**(val)

**LV\_STYLE\_CONST\_TEXT\_OUTLINE\_STROKE\_COLOR**(val)  
**LV\_STYLE\_CONST\_TEXT\_OUTLINE\_STROKE\_WIDTH**(val)  
**LV\_STYLE\_CONST\_TEXT\_OUTLINE\_STROKE\_OPA**(val)  
**LV\_STYLE\_CONST\_RADIUS**(val)  
**LV\_STYLE\_CONST\_RADIAL\_OFFSET**(val)  
**LV\_STYLE\_CONST\_CLIP\_CORNER**(val)  
**LV\_STYLE\_CONST\_OPA**(val)  
**LV\_STYLE\_CONST\_OPA\_LAYERED**(val)  
**LV\_STYLE\_CONST\_COLOR\_FILTER\_DSC**(val)  
**LV\_STYLE\_CONST\_COLOR\_FILTER\_OPA**(val)  
**LV\_STYLE\_CONST\_RECOLOR**(val)  
**LV\_STYLE\_CONST\_RECOLOR\_OPA**(val)  
**LV\_STYLE\_CONST\_ANIM**(val)  
**LV\_STYLE\_CONST\_ANIM\_DURATION**(val)  
**LV\_STYLE\_CONST\_TRANSITION**(val)  
**LV\_STYLE\_CONST\_BLEND\_MODE**(val)  
**LV\_STYLE\_CONST\_LAYOUT**(val)  
**LV\_STYLE\_CONST\_BASE\_DIR**(val)  
**LV\_STYLE\_CONST\_BITMAP\_MASK\_SRC**(val)  
**LV\_STYLE\_CONST\_ROTARY\_SENSITIVITY**(val)

## Functions

```

void lv_style_set_width(lv_style_t *style, int32_t value)
void lv_style_set_min_width(lv_style_t *style, int32_t value)
void lv_style_set_max_width(lv_style_t *style, int32_t value)
void lv_style_set_height(lv_style_t *style, int32_t value)
void lv_style_set_min_height(lv_style_t *style, int32_t value)
void lv_style_set_max_height(lv_style_t *style, int32_t value)
void lv_style_set_length(lv_style_t *style, int32_t value)
void lv_style_set_x(lv_style_t *style, int32_t value)
void lv_style_set_y(lv_style_t *style, int32_t value)
void lv_style_set_align(lv_style_t *style, lv_align_t value)
void lv_style_set_transform_width(lv_style_t *style, int32_t value)
void lv_style_set_transform_height(lv_style_t *style, int32_t value)

```

```
void lv_style_set_translate_x(lv_style_t *style, int32_t value)
void lv_style_set_translate_y(lv_style_t *style, int32_t value)
void lv_style_set_translate_radial(lv_style_t *style, int32_t value)
void lv_style_set_transform_scale_x(lv_style_t *style, int32_t value)
void lv_style_set_transform_scale_y(lv_style_t *style, int32_t value)
void lv_style_set_transform_rotation(lv_style_t *style, int32_t value)
void lv_style_set_transform_pivot_x(lv_style_t *style, int32_t value)
void lv_style_set_transform_pivot_y(lv_style_t *style, int32_t value)
void lv_style_set_transform_skew_x(lv_style_t *style, int32_t value)
void lv_style_set_transform_skew_y(lv_style_t *style, int32_t value)
void lv_style_set_pad_top(lv_style_t *style, int32_t value)
void lv_style_set_pad_bottom(lv_style_t *style, int32_t value)
void lv_style_set_pad_left(lv_style_t *style, int32_t value)
void lv_style_set_pad_right(lv_style_t *style, int32_t value)
void lv_style_set_pad_row(lv_style_t *style, int32_t value)
void lv_style_set_pad_column(lv_style_t *style, int32_t value)
void lv_style_set_pad_radial(lv_style_t *style, int32_t value)
void lv_style_set_margin_top(lv_style_t *style, int32_t value)
void lv_style_set_margin_bottom(lv_style_t *style, int32_t value)
void lv_style_set_margin_left(lv_style_t *style, int32_t value)
void lv_style_set_margin_right(lv_style_t *style, int32_t value)
void lv_style_set_bg_color(lv_style_t *style, lv_color_t value)
void lv_style_set_bg_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_bg_grad_color(lv_style_t *style, lv_color_t value)
void lv_style_set_bg_grad_dir(lv_style_t *style, lv_grad_dir_t value)
void lv_style_set_bg_main_stop(lv_style_t *style, int32_t value)
void lv_style_set_bg_grad_stop(lv_style_t *style, int32_t value)
void lv_style_set_bg_main_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_bg_grad_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_bg_grad(lv_style_t *style, const lv_grad_dsc_t *value)
void lv_style_set_bg_image_src(lv_style_t *style, const void *value)
void lv_style_set_bg_image_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_bg_image_recolor(lv_style_t *style, lv_color_t value)
```

```
void lv_style_set_bg_image_recolor_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_bg_image_tiled(lv_style_t *style, bool value)
void lv_style_set_border_color(lv_style_t *style, lv_color_t value)
void lv_style_set_border_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_border_width(lv_style_t *style, int32_t value)
void lv_style_set_border_side(lv_style_t *style, lv_border_side_t value)
void lv_style_set_border_post(lv_style_t *style, bool value)
void lv_style_set_outline_width(lv_style_t *style, int32_t value)
void lv_style_set_outline_color(lv_style_t *style, lv_color_t value)
void lv_style_set_outline_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_outline_pad(lv_style_t *style, int32_t value)
void lv_style_set_shadow_width(lv_style_t *style, int32_t value)
void lv_style_set_shadow_offset_x(lv_style_t *style, int32_t value)
void lv_style_set_shadow_offset_y(lv_style_t *style, int32_t value)
void lv_style_set_shadow_spread(lv_style_t *style, int32_t value)
void lv_style_set_shadow_color(lv_style_t *style, lv_color_t value)
void lv_style_set_shadow_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_image_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_image_recolor(lv_style_t *style, lv_color_t value)
void lv_style_set_image_recolor_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_line_width(lv_style_t *style, int32_t value)
void lv_style_set_line_dash_width(lv_style_t *style, int32_t value)
void lv_style_set_line_dash_gap(lv_style_t *style, int32_t value)
void lv_style_set_line_rounded(lv_style_t *style, bool value)
void lv_style_set_line_color(lv_style_t *style, lv_color_t value)
void lv_style_set_line_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_arc_width(lv_style_t *style, int32_t value)
void lv_style_set_arc_rounded(lv_style_t *style, bool value)
void lv_style_set_arc_color(lv_style_t *style, lv_color_t value)
void lv_style_set_arc_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_arc_image_src(lv_style_t *style, const void *value)
void lv_style_set_text_color(lv_style_t *style, lv_color_t value)
void lv_style_set_text_opa(lv_style_t *style, lv_opa_t value)
```

```

void lv_style_set_text_font(lv_style_t *style, const lv_font_t *value)
void lv_style_set_text_letter_space(lv_style_t *style, int32_t value)
void lv_style_set_text_line_space(lv_style_t *style, int32_t value)
void lv_style_set_text_decor(lv_style_t *style, lv_text_decor_t value)
void lv_style_set_text_align(lv_style_t *style, lv_text_align_t value)
void lv_style_set_text_outline_stroke_color(lv_style_t *style, lv_color_t value)
void lv_style_set_text_outline_stroke_width(lv_style_t *style, int32_t value)
void lv_style_set_text_outline_stroke_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_radius(lv_style_t *style, int32_t value)
void lv_style_set_radial_offset(lv_style_t *style, int32_t value)
void lv_style_set_clip_corner(lv_style_t *style, bool value)
void lv_style_set_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_opa_layered(lv_style_t *style, lv_opa_t value)
void lv_style_set_color_filter_dsc(lv_style_t *style, const lv_color_filter_dsc_t *value)
void lv_style_set_color_filter_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_recolor(lv_style_t *style, lv_color_t value)
void lv_style_set_recolor_opa(lv_style_t *style, lv_opa_t value)
void lv_style_set_anim(lv_style_t *style, const lv_anim_t *value)
void lv_style_set_anim_duration(lv_style_t *style, uint32_t value)
void lv_style_set_transition(lv_style_t *style, const lv_style_transition_desc_t *value)
void lv_style_set_blend_mode(lv_style_t *style, lv_blend_mode_t value)
void lv_style_set_layout(lv_style_t *style, uint16_t value)
void lv_style_set_base_dir(lv_style_t *style, lv_base_dir_t value)
void lv_style_set_bitmap_mask_src(lv_style_t *style, const void *value)
void lv_style_set_rotary_sensitivity(lv_style_t *style, uint32_t value)

```

#### 14.18.35 *lv\_style\_private.h*

#### 14.18.36 *lv\_temp.h*

#### 14.18.37 *lv\_text.h*

#### Defines

**LV\_TXT\_ENC\_UTF8**

**LV\_TXT\_ENC\_ASCII**

**LV\_TEXT\_LEN\_MAX**

**Enums****enum `lv_text_flag_t`**

Options for text rendering.

*Values:*

enumerator **LV\_TEXT\_FLAG\_NONE**

enumerator **LV\_TEXT\_FLAG\_EXPAND**

enumerator **LV\_TEXT\_FLAG\_FIT**

Max-width is already equal to the longest line. (Used to skip some calculation)

enumerator **LV\_TEXT\_FLAG\_BREAK\_ALL**

To prevent overflow, insert breaks between any two characters. Otherwise breaks are inserted at word boundaries, as configured via LV\_TXT\_BREAK\_CHARS or according to LV\_TXT\_LINE\_BREAK\_LONG\_LEN, LV\_TXT\_LINE\_BREAK\_LONG\_PRE\_MIN\_LEN, and LV\_TXT\_LINE\_BREAK\_LONG\_POST\_MIN\_LEN.

enumerator **LV\_TEXT\_FLAG\_RECOLOR**

Enable parsing of recolor command

**enum `lv_text_align_t`**

Label align policy

*Values:*

enumerator **LV\_TEXT\_ALIGN\_AUTO**

Align text auto

enumerator **LV\_TEXT\_ALIGN\_LEFT**

Align text to left

enumerator **LV\_TEXT\_ALIGN\_CENTER**

Align text to center

enumerator **LV\_TEXT\_ALIGN\_RIGHT**

Align text to right

**enum `lv_text_cmd_state_t`**

State machine for text renderer.

*Values:*

enumerator **LV\_TEXT\_CMD\_STATE\_WAIT**

Waiting for command

enumerator **LV\_TEXT\_CMD\_STATE\_PAR**

Processing the parameter

enumerator **LV\_TEXT\_CMD\_STATE\_IN**

Processing the command

## Functions

`void lv_text_attributes_init(lv_text_attributes_t *attributes)`

Initialize the text attributes descriptor

### Parametri

**attributes** -- the text attributes descriptor to initialize

`void lv_text_get_size(lv_point_t *size_res, const char *text, const lv_font_t *font, lv_text_attributes_t *attributes)`

Get size of a text

### Parametri

- **size\_res** -- pointer to a 'point\_t' variable to store the result
- **text** -- pointer to a text
- **font** -- pointer to font of the text
- **attributes** -- the text attributes, flags for line break behaviour, spacing etc

struct **lv\_text\_attributes\_t**

## Public Members

`int32_t letter_space`

Letter space between letters

`int32_t line_space`

Space between lines of text

`int32_t max_width`

Max width of the text (break the lines to fit this size). Set COORD\_MAX to avoid

`lv_text_flag_t text_flags`

## 14.18.38 lv\_text\_ap.h

### Defines

**LV\_UNDEF\_ARABIC\_PERSIAN\_CHARS**

**LV\_AP\_ALPHABET\_BASE\_CODE**

**LV\_AP\_END\_CHARS\_LIST**

## Functions

`uint32_t lv_text_ap_calc_bytes_count(const char *txt)`

`void lv_text_ap_proc(const char *txt, char *txt_out)`

## 14.18.39 lv\_text\_private.h

### Functions

`int32_t lv_text_get_width(const char *txt, uint32_t length, const lv_font_t *font, const lv_text_attributes_t *attributes)`

Give the length of a text with a given font with text flags



- **txt** -- pointer to '\0' terminated string
- **letter** -- the first decoded Unicode character or 0 on invalid data code
- **letter\_next** -- the second decoded Unicode character or 0 on invalid data code
- **ofs** -- start index in 'txt' where to start. After the call it will point to the next encoded char in 'txt'. NULL to use txt[0] as index

static inline bool **lv\_text\_is\_break\_char**(uint32\_t letter)

Test if char is break char or not (a text can broken here or not)

**Parametri**

**letter** -- a letter

**Ritorna**

    false: 'letter' is not break char

static inline bool **lv\_text\_is\_a\_word**(uint32\_t letter)

Test if char is break char or not (a text can broken here or not)

**Parametri**

**letter** -- a letter

**Ritorna**

    false: 'letter' is not break char

static inline bool **lv\_text\_is\_marker**(uint32\_t letter)

Test if character can be treated as marker, and don't need to be rendered. Note, this is not a full list. Add your findings to the list.

**Parametri**

**letter** -- a letter

**Ritorna**

    true if so

## Variables

uint8\_t (\*const **lv\_text\_encoded\_size**)(const char \*txt)

Give the size of an encoded character

**Param txt**

    pointer to a character in a string

**Return**

    length of the encoded character (1,2,3 ...). 0 in invalid

uint32\_t (\*const **lv\_text\_unicode\_to\_encoded**)(uint32\_t letter\_uni)

Convert a Unicode letter to encoded

**Param letter\_uni**

    a Unicode letter

**Return**

    Encoded character in Little Endian to be compatible with C chars (e.g. 'Á', 'Ü')

uint32\_t (\*const **lv\_text\_encoded\_conv\_wc**)(uint32\_t c)

Convert a wide character, e.g. 'Á' little endian to be compatible with the encoded format.

**Param c**

    a wide character

**Return**

    C in the encoded format

uint32\_t (\*const **lv\_text\_encoded\_next**)(const char \*txt, uint32\_t \*i\_start)

Decode the next encoded character from a string.

**Param txt**

    pointer to '\0' terminated string

**Param i\_start**

start index in 'txt' where to start. After the call it will point to the next encoded char in 'txt'.  
NULL to use txt[0] as index

**Return**

the decoded Unicode character or 0 on invalid data code

```
uint32_t (*const lv_text_encoded_prev)(const char *txt, uint32_t *i_start)
```

Get the previous encoded character form a string.

**Param txt**

pointer to '\0' terminated string

**Param i\_start**

index in 'txt' where to start. After the call it will point to the previous encoded char in 'txt'.

**Return**

the decoded Unicode character or 0 on invalid data

```
uint32_t (*const lv_text_encoded_get_byte_id)(const char *txt, uint32_t utf8_id)
```

Convert a letter index (in the encoded text) to byte index. E.g. in UTF-8 "AÁRT" index of 'R' is 2 but start at byte 3 because 'Á' is 2 bytes long

**Param txt**

a '\0' terminated UTF-8 string

**Param utf8\_id**

character index

**Return**

byte index of the 'enc\_id'th letter

```
uint32_t (*const lv_text_encoded_get_char_id)(const char *txt, uint32_t byte_id)
```

Convert a byte index (in an encoded text) to character index. E.g. in UTF-8 "AÁRT" index of 'R' is 2 but start at byte 3 because 'Á' is 2 bytes long

**Param txt**

a '\0' terminated UTF-8 string

**Param byte\_id**

byte index

**Return**

character index of the letter at 'byte\_id'th position

```
uint32_t (*const lv_text_get_encoded_length)(const char *txt)
```

Get the number of characters (and NOT bytes) in a string. E.g. in UTF-8 "ÁBC" is 3 characters (but 4 bytes)

**Param txt**

a '\0' terminated char string

**Return**

number of characters

## 14.18.40 lv\_timer.h

### Defines

**LV\_NO\_TIMER\_READY**

### Typedef

```
typedef void (*lv_timer_cb_t)(lv_timer_t*)
```

Timers execute this type of functions.

`typedef void (*lv_timer_handler_resume_cb_t)(void *data)`

Timer handler resume this type of function.

## Functions

`uint32_t lv_timer_handler(void)`

Call it periodically to handle lv\_timers.

**Ritorna**

time till it needs to be run next (in ms)

`uint32_t lv_timer_handler_run_in_period(uint32_t period)`

Call it in the super-loop of main() or threads. It will run `lv_timer_handler()` with a given period in ms. You can use it with sleep or delay in OS environment. This function is used to simplify the porting.

**Parametri**

- **period** -- the period for running `lv_timer_handler()`

**Ritorna**

the time after which it must be called again

`void lv_timer_periodic_handler(void)`

Call it in the super-loop of main() or threads. It will automatically call `lv_timer_handler()` at the right time.

This function is used to simplify the porting.

`void lv_timer_handler_set_resume_cb(lv_timer_handler_resume_cb_t cb, void *data)`

Set the resume callback to the timer handler

**Parametri**

- **cb** -- the function to call when timer handler is resumed
- **data** -- pointer to a resume data

`lv_timer_t *lv_timer_create_basic(void)`

Create an "empty" timer. It needs to be initialized with at least `lv_timer_set_cb` and `lv_timer_set_period`

**Ritorna**

pointer to the created timer

`lv_timer_t *lv_timer_create(lv_timer_cb_t timer_xcb, uint32_t period, void *user_data)`

Create a new lv\_timer

**Parametri**

- **timer\_xcb** -- a callback to call periodically. (the 'x' in the argument name indicates that it's not a fully generic function because it not follows the `func_name(object, callback, ...)` convention)
- **period** -- call period in ms unit
- **user\_data** -- custom parameter

**Ritorna**

pointer to the new timer

`void lv_timer_delete(lv_timer_t *timer)`

Delete a lv\_timer

**Parametri**

- **timer** -- pointer to an lv\_timer

`void lv_timer_pause(lv_timer_t *timer)`

Pause a timer.

**Parametri**

- **timer** -- pointer to an lv\_timer

`void lv_timer_resume(lv_timer_t *timer)`

Resume a timer.

**Parametri**

- **timer** -- pointer to an lv\_timer

**void lv\_timer\_set\_cb(lv\_timer\_t \*timer, *lv\_timer\_cb\_t* timer\_cb)**

Set the callback to the timer (the function to call periodically)

**Parametri**

- **timer** -- pointer to a timer
- **timer\_cb** -- the function to call periodically

**void lv\_timer\_set\_period(lv\_timer\_t \*timer, uint32\_t period)**

Set new period for a lv\_timer

**Parametri**

- **timer** -- pointer to a lv\_timer
- **period** -- the new period

**void lv\_timer\_ready(lv\_timer\_t \*timer)**

Make a lv\_timer ready. It will not wait its period.

**Parametri**

**timer** -- pointer to a lv\_timer.

**void lv\_timer\_set\_repeat\_count(lv\_timer\_t \*timer, int32\_t repeat\_count)**

Set the number of times a timer will repeat.

**Parametri**

- **timer** -- pointer to a lv\_timer.
- **repeat\_count** -- -1 : infinity; 0 : stop ; n>0: residual times

**void lv\_timer\_set\_auto\_delete(lv\_timer\_t \*timer, bool auto\_delete)**

Set whether a lv\_timer will be deleted automatically when it is called **repeat\_count** times.

**Parametri**

- **timer** -- pointer to a lv\_timer.
- **auto\_delete** -- true: auto delete; false: timer will be paused when it is called **repeat\_count** times.

**void lv\_timer\_set\_user\_data(lv\_timer\_t \*timer, void \*user\_data)**

Set custom parameter to the lv\_timer.

**Parametri**

- **timer** -- pointer to a lv\_timer.
- **user\_data** -- custom parameter

**void lv\_timer\_reset(lv\_timer\_t \*timer)**

Reset a lv\_timer. It will be called the previously set period milliseconds later.

**Parametri**

**timer** -- pointer to a lv\_timer.

**void lv\_timer\_enable(bool en)**

Enable or disable the whole lv\_timer handling

**Parametri**

**en** -- true: lv\_timer handling is running, false: lv\_timer handling is suspended

**uint32\_t lv\_timer\_get\_idle(void)**

Get idle percentage

**Ritorna**

the lv\_timer idle in percentage

**uint32\_t lv\_timer\_get\_time\_until\_next(void)**

Get the time remaining until the next timer will run

**Ritorna**

the time remaining in ms

**lv\_timer\_t \*lv\_timer\_get\_next(lv\_timer\_t \*timer)**

Iterate through the timers

**Parametri**

**timer** -- NULL to start iteration or the previous return value to get the next timer

**Ritorna**

the next timer or NULL if there is no more timer

**void \*lv\_timer\_get\_user\_data(lv\_timer\_t \*timer)**

Get the user\_data passed when the timer was created

**Parametri**

**timer** -- pointer to the lv\_timer

**Ritorna**

pointer to the user\_data

**bool lv\_timer\_get\_paused(lv\_timer\_t \*timer)**

Get the pause state of a timer

**Parametri**

**timer** -- pointer to a lv\_timer

**Ritorna**

true: timer is paused; false: timer is running

## 14.18.41 lv\_timer\_private.h

### Functions

**void lv\_timer\_core\_init(void)**

Init the lv\_timer module

**void lv\_timer\_core\_deinit(void)**

Deinit the lv\_timer module

**struct \_lv\_timer\_t**

#include <lv\_timer\_private.h> Descriptor of a lv\_timer

### Public Members

**uint32\_t period**

How often the timer should run

**uint32\_t last\_run**

Last time the timer ran

**lv\_timer\_cb\_t timer\_cb**

Timer function

**void \*user\_data**

Custom user data

**int32\_t repeat\_count**

1: One time; -1 : infinity; n>0: residual times

**uint32\_t paused**

**uint32\_t auto\_delete**

**struct lv\_timer\_state\_t**

**Public Members*****lv\_ll\_t timer\_ll***

Linked list to store the lv\_timers

**bool lv\_timer\_run****uint8\_t idle\_last****bool timer\_deleted****bool timer\_created****uint32\_t timer\_time\_until\_next****bool already\_running****uint32\_t periodic\_last\_tick****uint32\_t busy\_time****uint32\_t idle\_period\_start****uint32\_t run\_cnt*****lv\_timer\_handler\_resume\_cb\_t resume\_cb*****void \*resume\_data****14.18.42 lv\_tree.h**

Tree. The tree nodes are dynamically allocated by the 'lv\_mem' module.

**Defines****LV\_TREE\_NODE(n)****Typedef****typedef struct \_lv\_tree\_class\_t lv\_tree\_class\_t****typedef struct \_lv\_tree\_node\_t lv\_tree\_node\_t****typedef void (\*lv\_tree\_constructor\_cb\_t)(const lv\_tree\_class\_t \*class\_p, lv\_tree\_node\_t \*node)****typedef void (\*lv\_tree\_destructor\_cb\_t)(const lv\_tree\_class\_t \*class\_p, lv\_tree\_node\_t \*node)****typedef uint8\_t lv\_tree\_walk\_mode\_t**

typedef bool (\***lv\_tree\_traverse\_cb\_t**)(const *lv\_tree\_node\_t* \*node, void \*user\_data)

typedef bool (\***lv\_tree\_before\_cb\_t**)(const *lv\_tree\_node\_t* \*node, void \*user\_data)

typedef void (\***lv\_tree\_after\_cb\_t**)(const *lv\_tree\_node\_t* \*node, void \*user\_data)

## Enums

enum **lv\_tree\_walk\_mode\_t**

*Values:*

enumerator **LV\_TREE\_WALK\_PRE\_ORDER**

enumerator **LV\_TREE\_WALK\_POST\_ORDER**

## Functions

*lv\_tree\_node\_t* \***lv\_tree\_node\_create**(const *lv\_tree\_class\_t* \*class\_p, *lv\_tree\_node\_t* \*parent)

Create a tree node.

### Parametri

- **class\_p** -- pointer to a class of the node
- **parent** -- pointer to the parent node (or NULL if it's the root node)

### Ritorna

pointer to the new node

void **lv\_tree\_node\_delete**(*lv\_tree\_node\_t* \*node)

Delete a tree node and all its children recursively.

### Parametri

**node** -- pointer to the node to delete

bool **lv\_tree\_walk**(const *lv\_tree\_node\_t* \*node, *lv\_tree\_walk\_mode\_t* mode, *lv\_tree\_traverse\_cb\_t* cb, *lv\_tree\_before\_cb\_t* bcb, *lv\_tree\_after\_cb\_t* acb, void \*user\_data)

Walk the tree recursively and call a callback function on each node.

### Parametri

- **node** -- pointer to the root node of the tree
- **mode** -- LV\_TREE\_WALK\_PRE\_ORDER or LV\_TREE\_WALK\_POST\_ORDER
- **cb** -- callback function to call on each node
- **bcb** -- callback function to call before visiting a node
- **acb** -- callback function to call after visiting a node
- **user\_data** -- user data to pass to the callback functions

### Ritorna

true: traversal is finished; false: traversal broken

## Variables

const *lv\_tree\_class\_t* **lv\_tree\_node\_class**

struct **\_lv\_tree\_class\_t**

#include <lv\_tree.h> Describe the common methods of every object. Similar to a C++ class.

**Public Members**

```
const lv_tree_class_t *base_class

uint32_t instance_size

lv_tree_constructor_cb_t constructor_cb

lv_tree_destructor_cb_t destructor_cb

struct _lv_tree_node_t
#include <lv_tree.h> Description of a tree node
```

**Public Members**

```
lv_tree_node_t *parent

lv_tree_node_t **children

uint32_t child_cnt

uint32_t child_cap

const lv_tree_class_t *class_p
```

**14.18.43 lv\_utils.h****Functions**

```
void *lv_utils_bsearch(const void *key, const void *base, size_t n, size_t size, int (*cmp)(const void *pRef, const void *pElement))
```

Searches base[0] to base[n - 1] for an item that matches \*key.

**Nota**

The function cmp must return negative if it's first argument (the search key) is less than its second (a table entry), zero if equal, and positive if greater.

**Nota**

Items in the array must be in ascending order.

**Parametri**

- **key** -- Pointer to item being searched for
- **base** -- Pointer to first element to search
- **n** -- Number of elements
- **size** -- Size of each element
- **cmp** -- Pointer to comparison function (see unicode\_list\_compare() as a comparison function example)

**Ritorna**

a pointer to a matching item, or NULL if none exists.

`lv_result_t lv_draw_buf_save_to_file(const lv_draw_buf_t *draw_buf, const char *path)`

Save a draw buf to a file

**Parametri**

- **draw\_buf** -- pointer to a draw buffer
- **path** -- path to the file to save

**Ritorna**

LV\_RESULT\_OK: success; LV\_RESULT\_INVALID: error

`static inline uint32_t lv_swap_bytes_32(uint32_t x)`

Reverse the order of the bytes in a 32-bit value.

**Parametri**

- **x** -- a 32-bit value.

**Ritorna**

the value **x** with reversed byte-order.

`static inline uint16_t lv_swap_bytes_16(uint16_t x)`

Reverse the order of the bytes in a 16-bit value.

**Parametri**

- **x** -- a 16-bit value.

**Ritorna**

the value **x** with reversed byte-order.

## 14.18.44 cache

`lv_cache.h`

### Functions

`lv_cache_t *lv_cache_create(const lv_cache_class_t *cache_class, size_t node_size, size_t max_size, lv_cache_ops_t ops)`

Create a cache object with the given parameters.

**Parametri**

- **cache\_class** -- The class of the cache. Currently only support one two builtin classes:
  - lv\_cache\_class\_lru\_rb\_count for LRU-based cache with count-based eviction policy.
  - lv\_cache\_class\_lru\_rb\_size for LRU-based cache with size-based eviction policy.
- **node\_size** -- The node size is the size of the data stored in the cache..
- **max\_size** -- The max size is the maximum amount of memory or count that the cache can hold.
  - lv\_cache\_class\_lru\_rb\_count: max\_size is the maximum count of nodes in the cache.
  - lv\_cache\_class\_lru\_rb\_size: max\_size is the maximum size of the cache in bytes.
- **ops** -- A set of operations that can be performed on the cache. See `lv_cache_ops_t` for details.

**Ritorna**

Returns a pointer to the created cache object on success, `NULL` on error.

`void lv_cache_destroy(lv_cache_t *cache, void *user_data)`

Destroy a cache object.

**Parametri**

- **cache** -- The cache object pointer to destroy.
- **user\_data** -- A user data pointer that will be passed to the free callback.

`lv_cache_entry_t *lv_cache_acquire(lv_cache_t *cache, const void *key, void *user_data)`

Acquire a cache entry with the given key. If entry not in cache, it will return `NULL` (not found). If the entry is found, it's priority will be changed by the cache's policy. And the `lv_cache_entry_t::ref_cnt` will be incremented.

#### Parametri

- **cache** -- The cache object pointer to acquire the entry.
- **key** -- The key of the entry to acquire.
- **user\_data** -- A user data pointer that will be passed to the create callback.

#### Ritorna

Returns a pointer to the acquired cache entry on success with `lv_cache_entry_t::ref_cnt` incremented, `NULL` on error.

`lv_cache_entry_t *lv_cache_acquire_or_create(lv_cache_t *cache, const void *key, void *user_data)`

Acquire a cache entry with the given key. If the entry is not in the cache, it will create a new entry with the given key. If the entry is found, it's priority will be changed by the cache's policy. And the `lv_cache_entry_t::ref_cnt` will be incremented. If you want to use this API to simplify the code, you should provide a `lv_cache_ops_t::create_cb` that creates a new entry with the given key. This API is a combination of `lv_cache_acquire()` and `lv_cache_add()`. The effect is the same as calling `lv_cache_acquire()` and `lv_cache_add()` separately. And the internal impact on cache is also consistent with these two APIs.

#### Parametri

- **cache** -- The cache object pointer to acquire the entry.
- **key** -- The key of the entry to acquire or create.
- **user\_data** -- A user data pointer that will be passed to the create callback.

#### Ritorna

Returns a pointer to the acquired or created cache entry on success with `lv_cache_entry_t::ref_cnt` incremented, `NULL` on error.

`lv_cache_entry_t *lv_cache_add(lv_cache_t *cache, const void *key, void *user_data)`

Add a new cache entry with the given key and data. If the cache is full, the cache's policy will be used to evict an entry.

#### Parametri

- **cache** -- The cache object pointer to add the entry.
- **key** -- The key of the entry to add.
- **user\_data** -- A user data pointer that will be passed to the create callback.

#### Ritorna

Returns a pointer to the added cache entry on success with `lv_cache_entry_t::ref_cnt` incremented, `NULL` on error.

`void lv_cache_release(lv_cache_t *cache, lv_cache_entry_t *entry, void *user_data)`

Release a cache entry. The `lv_cache_entry_t::ref_cnt` will be decremented. If the `lv_cache_entry_t::ref_cnt` is zero, it will issue an error. If the entry passed to this function is the last reference to the data and the entry is marked as invalid, the cache's policy will be used to evict the entry.

#### Parametri

- **cache** -- The cache object pointer to release the entry.
- **entry** -- The cache entry pointer to release.
- **user\_data** -- A user data pointer that will be passed to the free callback.

`void lv_cache_reserve(lv_cache_t *cache, uint32_t reserved_size, void *user_data)`

Reserve a certain amount of memory/count in the cache. This function is useful when you want to reserve a certain amount of memory/count in advance, for example, when you know that you will need it later. When the current cache size is max than the reserved size, the function will evict entries until the reserved size is reached.

#### Parametri

- **cache** -- The cache object pointer to reserve.
- **reserved\_size** -- The amount of memory/count to reserve.
- **user\_data** -- A user data pointer that will be passed to the free callback.

`void lv_cache_drop(lv_cache_t *cache, const void *key, void *user_data)`

Drop a cache entry with the given key. If the entry is not in the cache, nothing will happen to it. If the entry is found, it will be removed from the cache and its data will be freed when the last reference to it is released.

#### Nota

The data will not be freed immediately but when the last reference to it is released. But this entry will not be found by `lv_cache_acquire()`. If you want cache a same key again, you should use `lv_cache_add()` or `lv_cache_acquire_or_create()`.

#### Parametri

- **cache** -- The cache object pointer to drop the entry.
- **key** -- The key of the entry to drop.
- **user\_data** -- A user data pointer that will be passed to the free callback.

`void lv_cache_drop_all(lv_cache_t *cache, void *user_data)`

Drop all cache entries. All entries will be removed from the cache and their data will be freed when the last reference to them is released.

#### Nota

If some entries are still referenced by other objects, it will issue an error. And this case shouldn't happen in normal cases..

#### Parametri

- **cache** -- The cache object pointer to drop all entries.
- **user\_data** -- A user data pointer that will be passed to the free callback.

`bool lv_cache_evict_one(lv_cache_t *cache, void *user_data)`

Evict one entry from the cache. The eviction policy will be used to select the entry to evict.

#### Parametri

- **cache** -- The cache object pointer to evict an entry.
- **user\_data** -- A user data pointer that will be passed to the free callback.

#### Ritorna

Returns true if an entry is evicted, false if no entry is evicted.

`void lv_cache_set_max_size(lv_cache_t *cache, size_t max_size, void *user_data)`

Set the maximum size of the cache. If the current cache size is greater than the new maximum size, the cache's policy will be used to evict entries until the new maximum size is reached. If set to 0, the cache will be disabled.

#### Nota

But this behavior will happen only new entries are added to the cache.

#### Parametri

- **cache** -- The cache object pointer to set the maximum size.
- **max\_size** -- The new maximum size of the cache.
- **user\_data** -- A user data pointer that will be passed to the free callback.

`size_t lv_cache_get_max_size(lv_cache_t *cache, void *user_data)`

Get the maximum size of the cache.

#### Parametri

- **cache** -- The cache object pointer to get the maximum size.
- **user\_data** -- A user data pointer that will be passed to the free callback.

#### Ritorna

Returns the maximum size of the cache.

`size_t lv_cache_get_size(lv_cache_t *cache, void *user_data)`

Get the current size of the cache.

**Parametri**

- **cache** -- The cache object pointer to get the current size.
- **user\_data** -- A user data pointer that will be passed to the free callback.

**Ritorna**

Returns the current size of the cache.

`size_t lv_cache_get_free_size(lv_cache_t *cache, void *user_data)`

Get the free size of the cache.

**Parametri**

- **cache** -- The cache object pointer to get the free size.
- **user\_data** -- A user data pointer that will be passed to the free callback.

**Ritorna**

Returns the free size of the cache.

`bool lv_cache_is_enabled(lv_cache_t *cache)`

Return true if the cache is enabled. Disabled cache means that when the max\_size of the cache is 0. In this case, all cache operations will be no-op.

**Parametri**

- **cache** -- The cache object pointer to check if it's disabled.

**Ritorna**

Returns true if the cache is enabled, false otherwise.

`void lv_cache_set_compare_cb(lv_cache_t *cache, lv_cache_compare_cb_t compare_cb, void *user_data)`

Set the compare callback of the cache.

**Parametri**

- **cache** -- The cache object pointer to set the compare callback.
- **compare\_cb** -- The compare callback to set.
- **user\_data** -- A user data pointer.

`void lv_cache_set_create_cb(lv_cache_t *cache, lv_cache_create_cb_t alloc_cb, void *user_data)`

Set the create callback of the cache.

**Parametri**

- **cache** -- The cache object pointer to set the create callback.
- **alloc\_cb** -- The create callback to set.
- **user\_data** -- A user data pointer.

`void lv_cache_set_free_cb(lv_cache_t *cache, lv_cache_free_cb_t free_cb, void *user_data)`

Set the free callback of the cache.

**Parametri**

- **cache** -- The cache object pointer to set the free callback.
- **free\_cb** -- The free callback to set.
- **user\_data** -- A user data pointer.

`void lv_cache_set_name(lv_cache_t *cache, const char *name)`

Give a name for a cache object. Only the pointer of the string is saved.

**Parametri**

- **cache** -- The cache object pointer to set the name.
- **name** -- The name of the cache.

`const char *lv_cache_get_name(lv_cache_t *cache)`

Get the name of a cache object.

**Parametri**

- **cache** -- The cache object pointer to get the name.

**Ritorna**

Returns the name of the cache.

---

```
lv_iter_t *lv_cache_iter_create(lv_cache_t *cache)
```

Create an iterator for the cache object. The iterator is used to iterate over all cache entries.

**Parametri**

- **cache** -- The cache object pointer to create the iterator.

**Ritorna**

- Returns a pointer to the created iterator on success, **NULL** on error.

## lv\_cache\_entry.h

### Functions

```
uint32_t lv_cache_entry_get_size(const uint32_t node_size)
```

Get the size of a cache entry.

**Parametri**

- **node\_size** -- The size of the node in the cache.

**Ritorna**

- The size of the cache entry.

```
int32_t lv_cache_entry_get_ref(lv_cache_entry_t *entry)
```

Get the reference count of a cache entry.

**Parametri**

- **entry** -- The cache entry to get the reference count of.

**Ritorna**

- The reference count of the cache entry.

```
uint32_t lv_cache_entry_get_node_size(lv_cache_entry_t *entry)
```

Get the node size of a cache entry. Which is the same size with *lv\_cache\_entry\_get\_size()*'s **node\_size** parameter.

**Parametri**

- **entry** -- The cache entry to get the node size of.

**Ritorna**

- The node size of the cache entry.

```
bool lv_cache_entry_is_invalid(lv_cache_entry_t *entry)
```

Check if a cache entry is invalid.

**Parametri**

- **entry** -- The cache entry to check.

**Ritorna**

- True: the cache entry is invalid. False: the cache entry is valid.

```
void *lv_cache_entry_get_data(lv_cache_entry_t *entry)
```

Get the data of a cache entry.

**Parametri**

- **entry** -- The cache entry to get the data of.

**Ritorna**

- The pointer to the data of the cache entry.

```
const lv_cache_t *lv_cache_entry_get_cache(const lv_cache_entry_t *entry)
```

Get the cache instance of a cache entry.

**Parametri**

- **entry** -- The cache entry to get the cache instance of.

**Ritorna**

- The pointer to the cache instance of the cache entry.

```
lv_cache_entry_t *lv_cache_entry_get_entry(void *data, const uint32_t node_size)
```

Get the cache entry of a data. The data should be allocated by the cache instance.

**Parametri**

- **data** -- The data to get the cache entry of.
- **node\_size** -- The size of the node in the cache.

**Ritorna**

The pointer to the cache entry of the data.

**lv\_cache\_entry\_t \*lv\_cache\_entry\_alloc**(const uint32\_t node\_size, const lv\_cache\_t \*cache)

Allocate a cache entry.

**Parametri**

- **node\_size** -- The size of the node in the cache.
- **cache** -- The cache instance to allocate the cache entry from.

**Ritorna**

The pointer to the allocated cache entry.

**void lv\_cache\_entry\_init(lv\_cache\_entry\_t \*entry, const lv\_cache\_t \*cache, const uint32\_t node\_size)**

Initialize a cache entry.

**Parametri**

- **entry** -- The cache entry to initialize.
- **cache** -- The cache instance to allocate the cache entry from.
- **node\_size** -- The size of the node in the cache.

**void lv\_cache\_entry\_delete(lv\_cache\_entry\_t \*entry)**

Deallocate a cache entry. And the data of the cache entry will be freed.

**Parametri**

**entry** -- The cache entry to deallocate.

**lv\_cache\_entry\_private.h**

**Defines**

**LV\_CACHE\_ENTRY\_FLAG\_INVALID**

**LV\_CACHE\_ENTRY\_FLAG\_DISABLE\_DELETE**

**LV\_CACHE\_ENTRY\_FLAG\_CLASS\_CUSTOM**

**Functions**

**void lv\_cache\_entry\_reset\_ref(lv\_cache\_entry\_t \*entry)**

**void lv\_cache\_entry\_inc\_ref(lv\_cache\_entry\_t \*entry)**

**void lv\_cache\_entry\_dec\_ref(lv\_cache\_entry\_t \*entry)**

**void lv\_cache\_entry\_set\_node\_size(lv\_cache\_entry\_t \*entry, uint32\_t node\_size)**

**void lv\_cache\_entry\_set\_cache(lv\_cache\_entry\_t \*entry, const lv\_cache\_t \*cache)**

**void \*lv\_cache\_entry\_acquire\_data(lv\_cache\_entry\_t \*entry)**

**void lv\_cache\_entry\_release\_data(lv\_cache\_entry\_t \*entry, void \*user\_data)**

**void lv\_cache\_entry\_set\_flag(lv\_cache\_entry\_t \*entry, uint8\_t flags)**

**void lv\_cache\_entry\_remove\_flag(lv\_cache\_entry\_t \*entry, uint8\_t flags)**

**bool lv\_cache\_entry\_has\_flag(lv\_cache\_entry\_t \*entry, uint8\_t flags)**

**struct \_lv\_cache\_entry\_t**

**Public Members**

```
const lv_cache_t *cache
```

```
int32_t ref_cnt
```

```
uint32_t node_size
```

```
uint8_t flags
```

**lv\_cache\_private.h**

**Typedef**

```
typedef struct _lv_cache_ops_t lv_cache_ops_t
```

```
typedef struct _lv_cache_class_t lv_cache_class_t
```

```
typedef int8_t lv_cache_compare_res_t
```

```
typedef bool (*lv_cache_create_cb_t)(void *node, void *user_data)
```

```
typedef void (*lv_cache_free_cb_t)(void *node, void *user_data)
```

```
typedef lv_cache_compare_res_t (*lv_cache_compare_cb_t)(const void *a, const void *b)
```

```
typedef void *(*lv_cache_alloc_cb_t)(void)
```

The cache instance allocation function, used by the cache class to allocate memory for cache instances.

**Return**

It should return a pointer to the allocated instance.

```
typedef bool (*lv_cache_init_cb_t)(lv_cache_t *cache)
```

The cache instance initialization function, used by the cache class to initialize the cache instance.

**Return**

It should return true if the initialization is successful, false otherwise.

```
typedef void (*lv_cache_destroy_cb_t)(lv_cache_t *cache, void *user_data)
```

The cache instance destruction function, used by the cache class to destroy the cache instance.

```
typedef lv_cache_entry_t *(*lv_cache_get_cb_t)(lv_cache_t *cache, const void *key, void *user_data)
```

The cache get function, used by the cache class to get a cache entry by its key.

**Return**

NULL if the key is not found.

```
typedef lv_cache_entry_t *(*lv_cache_add_cb_t)(lv_cache_t *cache, const void *key, void *user_data)
```

The cache add function, used by the cache class to add a cache entry with a given key. This function only cares about how to add the entry, it doesn't check if the entry already exists and doesn't care about is it a victim or not.

**Return**

the added cache entry, or NULL if the entry is not added.

typedef void (\***lv\_cache\_remove\_cb\_t**)(lv\_cache\_t \*cache, lv\_cache\_entry\_t \*entry, void \*user\_data)

The cache remove function, used by the cache class to remove a cache entry from the cache but doesn't free the memory.. This function only cares about how to remove the entry, it doesn't care about is it a victim or not.

typedef void (\***lv\_cache\_drop\_cb\_t**)(lv\_cache\_t \*cache, const void \*key, void \*user\_data)

The cache drop function, used by the cache class to remove a cache entry from the cache and free the memory.

typedef void (\***lv\_cache\_drop\_all\_cb\_t**)(lv\_cache\_t \*cache, void \*user\_data)

The cache drop all function, used by the cache class to remove all cache entries from the cache and free the memory.

typedef lv\_cache\_entry\_t \*(\***lv\_cache\_get\_victim\_cb**)(lv\_cache\_t \*cache, void \*user\_data)

The cache get victim function, used by the cache class to get a victim entry to be evicted.

typedef *lv\_cache\_reserve\_cond\_res\_t* (\***lv\_cache\_reserve\_cond\_cb**)(lv\_cache\_t \*cache, const void \*key, size\_t size, void \*user\_data)

The cache reserve condition function, used by the cache class to check if a new entry can be added to the cache without exceeding its maximum size. See *lv\_cache\_reserve\_cond\_res\_t* for the possible results.

typedef lv\_iter\_t \*(\***lv\_cache\_iter\_create\_cb**)(lv\_cache\_t \*cache)

The cache iterator creation function, used by the cache class to create an iterator for the cache.

#### Return

A pointer to the created iterator, or NULL if the iterator cannot be created.

typedef struct *lv\_cache\_slot\_size\_t* **lv\_cache\_slot\_size\_t**

## Enums

enum **lv\_cache\_reserve\_cond\_res\_t**

The result of the cache reserve condition callback

*Values:*

enumerator **LV\_CACHE\_RESERVE\_COND\_OK**

The condition is met and no entries need to be evicted

enumerator **LV\_CACHE\_RESERVE\_COND\_TOO\_LARGE**

The condition is not met and the reserve size is too large

enumerator **LV\_CACHE\_RESERVE\_COND\_NEED\_VICTIM**

The condition is not met and a victim is needed to be evicted

enumerator **LV\_CACHE\_RESERVE\_COND\_ERROR**

An error occurred while checking the condition

struct **\_lv\_cache\_ops\_t**

#include <lv\_cache\_private.h> The cache operations struct

**Public Members*****lv\_cache\_compare\_cb\_t compare\_cb***

Compare function for keys

***lv\_cache\_create\_cb\_t create\_cb***

Create function for nodes

***lv\_cache\_free\_cb\_t free\_cb***

Free function for nodes

**struct *\_lv\_cache\_t***

#include &lt;lv\_cache\_private.h&gt; The cache entry struct

**Public Members****const *lv\_cache\_class\_t* \*clz**

Cache class. There are two built-in classes:

- lv\_cache\_class\_lru\_rb\_count for LRU-based cache with count-based eviction policy.
- lv\_cache\_class\_lru\_rb\_size for LRU-based cache with size-based eviction policy.

**uint32\_t node\_size**

Size of a node

**uint32\_t max\_size**

Maximum size of the cache

**uint32\_t size**

Current size of the cache

***lv\_cache\_ops\_t ops***Cache operations struct *\_lv\_cache\_ops\_t****lv\_mutex\_t lock***

Cache lock used to protect the cache in multithreading environments

**const char \*name**

Name of the cache

**struct *\_lv\_cache\_class\_t***

#include &lt;lv\_cache\_private.h&gt; Cache class struct for building custom cache classes

Examples:

- lv\_cache\_class\_lru\_rb\_count for LRU-based cache with count-based eviction policy.
- lv\_cache\_class\_lru\_rb\_size for LRU-based cache with size-based eviction policy.

**Public Members*****lv\_cache\_alloc\_cb\_t alloc\_cb***

The allocation function for cache entries

***lv\_cache\_init\_cb\_t* init\_cb**

The initialization function for cache entries

***lv\_cache\_destroy\_cb\_t* destroy\_cb**

The destruction function for cache entries

***lv\_cache\_get\_cb\_t* get\_cb**

The get function for cache entries

***lv\_cache\_add\_cb\_t* add\_cb**

The add function for cache entries

***lv\_cache\_remove\_cb\_t* remove\_cb**

The remove function for cache entries

***lv\_cache\_drop\_cb\_t* drop\_cb**

The drop function for cache entries

***lv\_cache\_drop\_all\_cb\_t* drop\_all\_cb**

The drop all function for cache entries

***lv\_cache\_get\_victim\_cb* get\_victim\_cb**

The get victim function for cache entries

***lv\_cache\_reserve\_cond\_cb* reserve\_cond\_cb**

The reserve condition function for cache entries

***lv\_cache\_iter\_create\_cb* iter\_create\_cb**

The iterator creation function for cache entries

struct **\_lv\_cache\_slot\_size\_t**

#include <lv\_cache\_private.h> Cache entry slot struct

To add new fields to the cache entry, add them to a new struct and add it to the first field of the cache data struct. And this one is a size slot for the cache entry.

## Public Members

size\_t **size**

**class**

**lv\_cache\_class.h**

**lv\_cache\_lru\_ll.h**

## Variables

const *lv\_cache\_class\_t* **lv\_cache\_class\_lru\_ll\_count**

const *lv\_cache\_class\_t* **lv\_cache\_class\_lru\_ll\_size**

**lv\_cache\_lru\_rb.h****Variables**

```
const lv_cache_class_t lv_cache_class_lru_rb_count
```

```
const lv_cache_class_t lv_cache_class_lru_rb_size
```

**lv\_cache\_sc\_da.h****Variables**

```
const lv_cache_class_t lv_cache_class_sc_da
```

**instance****lv\_cache\_instance.h****lv\_image\_cache.h****Functions**

`lv_result_t lv_image_cache_init(uint32_t size)`

Initialize image cache.

**Parametri**

- **size** -- size of the cache in bytes.

**Ritorna**

LV\_RESULT\_OK: initialization succeeded, LV\_RESULT\_INVALID: failed.

`void lv_image_cache_resize(uint32_t new_size, bool evict_now)`

Resize image cache. If set to 0, the cache will be disabled.

**Parametri**

- **new\_size** -- new size of the cache in bytes.
- **evict\_now** -- true: evict the images should be removed by the eviction policy, false: wait for the next cache cleanup.

`void lv_image_cache_drop(const void *src)`

Invalidate image cache. Use NULL to invalidate all images.

**Parametri**

- **src** -- pointer to an image source.

`bool lv_image_cache_is_enabled(void)`

Return true if the image cache is enabled.

**Ritorna**

- true: enabled, false: disabled.

`lv_iter_t *lv_image_cache_iter_create(void)`

Create an iterator to iterate over the image cache.

**Ritorna**

- an iterator to iterate over the image cache.

`void lv_image_cache_dump(void)`

Dump the content of the image cache in a human-readable format with cache order.

**lv\_image\_header\_cache.h****Functions****lv\_result\_t lv\_image\_header\_cache\_init(uint32\_t count)**

Initialize image header cache.

**Parametri**• **count** -- initial size of the cache in count of image headers.**Ritorna**

LV\_RESULT\_OK: initialization succeeded, LV\_RESULT\_INVALID: failed.

**void lv\_image\_header\_cache\_resize(uint32\_t count, bool evict\_now)**

Resize image header cache. If set to 0, the cache is disabled.

**Parametri**

- **count** -- new max count of cached image headers.
- **evict\_now** -- true: evict the image headers should be removed by the eviction policy, false: wait for the next cache cleanup.

**void lv\_image\_header\_cache\_drop(const void \*src)**

Invalidate image header cache. Use NULL to invalidate all image headers. It's also automatically called when an image is invalidated.

**Parametri**• **src** -- pointer to an image source.**bool lv\_image\_header\_cache\_is\_enabled(void)**

Return true if the image header cache is enabled.

**Ritorna**

true: enabled, false: disabled.

**lv\_iter\_t \*lv\_image\_header\_cache\_iter\_create(void)**

Create an iterator to iterate over the image header cache.

**Ritorna**

an iterator to iterate over the image header cache.

**void lv\_image\_header\_cache\_dump(void)**

Dump the content of the image header cache in a human-readable format with cache order.

## 14.19 osal

### 14.19.1 lv\_linux\_private.h

**Defines****LV\_PROC\_STAT\_PARAMS\_LEN****Functions****uint32\_t lv\_os\_get\_proc\_idle\_percent(void)****union lv\_proc\_stat\_t****Public Members**  **uint32\_t user**  **uint32\_t nice**

```

uint32_t system

uint32_t idle

uint32_t irq

uint32_t softirq

uint32_t steal

struct lv_proc_stat_t fields

uint32_t buffer[7]

```

## 14.19.2 lv\_os.h

### Enums

#### enum **lv\_thread\_prio\_t**

*Values:*

enumerator **LV\_THREAD\_PRIO\_LOWEST**

enumerator **LV\_THREAD\_PRIO\_LOW**

enumerator **LV\_THREAD\_PRIO\_MID**

enumerator **LV\_THREAD\_PRIO\_HIGH**

enumerator **LV\_THREAD\_PRIO\_HIGHEST**

### Functions

#### uint32\_t **lv\_os\_get\_idle\_percent**(void)

Set it for **LV\_SYSMON\_GET\_IDLE** to show the CPU usage

**Ritorna**

the idle percentage since the last call

static inline lv\_result\_t **lv\_thread\_init**(*lv\_thread\_t* \*thread, const char \*const name, *lv\_thread\_prio\_t* prio,  
void (\*callback)(void\*), size\_t stack\_size, void \*user\_data)

static inline lv\_result\_t **lv\_thread\_delete**(*lv\_thread\_t* \*thread)

static inline lv\_result\_t **lv\_mutex\_init**(*lv\_mutex\_t* \*mutex)

static inline lv\_result\_t **lv\_mutex\_lock**(*lv\_mutex\_t* \*mutex)

static inline lv\_result\_t **lv\_mutex\_lock\_isr**(*lv\_mutex\_t* \*mutex)

static inline lv\_result\_t **lv\_mutex\_unlock**(*lv\_mutex\_t* \*mutex)

static inline lv\_result\_t **lv\_mutex\_delete**(*lv\_mutex\_t* \*mutex)

```
static inline lv_result_t lv_thread_sync_init(lv_thread_sync_t *sync)
static inline lv_result_t lv_thread_sync_wait(lv_thread_sync_t *sync)
static inline lv_result_t lv_thread_sync_signal(lv_thread_sync_t *sync)
static inline lv_result_t lv_thread_sync_signal_isr(lv_thread_sync_t *sync)
static inline lv_result_t lv_thread_sync_delete(lv_thread_sync_t *sync)
static inline void lv_lock(void)
static inline lv_result_t lv_lock_isr(void)
static inline void lv_unlock(void)
```

### 14.19.3 lv\_os\_none.h

#### Typedef

typedef int **lv\_mutex\_t**

typedef int **lv\_thread\_t**

typedef int **lv\_thread\_sync\_t**

### 14.19.4 lv\_os\_private.h

#### Functions

void **lv\_os\_init**(void)

Initialize the OS layer

## 14.20 others

### 14.20.1 file\_explorer

**lv\_file\_explorer.h**

#### Enums

enum **lv\_file\_explorer\_sort\_t**

*Values:*

enumerator **LV\_EXPLORER\_SORT\_NONE**

enumerator **LV\_EXPLORER\_SORT\_KIND**

enum **lv\_file\_explorer\_dir\_t**

*Values:*

enumerator **LV\_EXPLORER\_HOME\_DIR**

enumerator **LV\_EXPLORER\_MUSIC\_DIR**

enumerator **LV\_EXPLORER\_PICTURES\_DIR**

enumerator **LV\_EXPLORER\_VIDEO\_DIR**

enumerator **LV\_EXPLORER\_DOCS\_DIR**

enumerator **LV\_EXPLORER\_FS\_DIR**

## Functions

`lv_obj_t *lv_file_explorer_create(lv_obj_t *parent)`

`void lv_file_explorer_set_quick_access_path(lv_obj_t *obj, lv_file_explorer_dir_t dir, const char *path)`

Set file\_explorer

### Parametri

- **obj** -- pointer to a label object
- **dir** -- the dir from 'lv\_file\_explorer\_dir\_t' enum.
- **path** -- path

`void lv_file_explorer_set_sort(lv_obj_t *obj, lv_file_explorer_sort_t sort)`

Set file\_explorer sort

### Parametri

- **obj** -- pointer to a file explorer object
- **sort** -- the sort from 'lv\_file\_explorer\_sort\_t' enum.

`const char *lv_file_explorer_get_selected_file_name(const lv_obj_t *obj)`

Get file explorer Selected file

### Parametri

**obj** -- pointer to a file explorer object

### Ritorna

    pointer to the file explorer selected file name

`const char *lv_file_explorer_get_current_path(const lv_obj_t *obj)`

Get file explorer cur path

### Parametri

**obj** -- pointer to a file explorer object

### Ritorna

    pointer to the file explorer cur path

`lv_obj_t *lv_file_explorer_get_file_table(lv_obj_t *obj)`

Get file explorer file list obj(lv\_table)

### Parametri

**obj** -- pointer to a file explorer object

### Ritorna

    pointer to the file explorer file table obj(lv\_table)

`lv_obj_t *lv_file_explorer_get_header(lv_obj_t *obj)`

Get file explorer head area obj

### Parametri

**obj** -- pointer to a file explorer object

### Ritorna

    pointer to the file explorer head area obj(lv\_obj)

`lv_obj_t *lv_file_explorer_get_path_label(lv_obj_t *obj)`

Get file explorer path obj(label)

### Parametri

**obj** -- pointer to a file explorer object

**Ritorna**  
pointer to the file explorer path obj(lv\_label)

**lv\_obj\_t \*lv\_file\_explorer\_get\_quick\_access\_area(lv\_obj\_t \*obj)**

Get file explorer head area obj

**Parametri**  
**obj** -- pointer to a file explorer object

**Ritorna**  
pointer to the file explorer quick access area obj(lv\_obj)

**lv\_obj\_t \*lv\_file\_explorer\_get\_places\_list(lv\_obj\_t \*obj)**

Get file explorer places list obj(lv\_list)

**Parametri**  
**obj** -- pointer to a file explorer object

**Ritorna**  
pointer to the file explorer places list obj(lv\_list)

**lv\_obj\_t \*lv\_file\_explorer\_get\_device\_list(lv\_obj\_t \*obj)**

Get file explorer device list obj(lv\_list)

**Parametri**  
**obj** -- pointer to a file explorer object

**Ritorna**  
pointer to the file explorer device list obj(lv\_list)

*lv\_file\_explorer\_sort\_t lv\_file\_explorer\_get\_sort(const lv\_obj\_t \*obj)*

Set file\_explorer sort

**Parametri**  
**obj** -- pointer to a file explorer object

**Ritorna**  
the current mode from 'lv\_file\_explorer\_sort\_t'

**void lv\_file\_explorer\_open\_dir(lv\_obj\_t \*obj, const char \*dir)**

Open a specified path

**Parametri**

- **obj** -- pointer to a file explorer object
- **dir** -- pointer to the path

## Variables

const lv\_obj\_class\_t **lv\_file\_explorer\_class**

**lv\_file\_explorer\_private.h**

## Enums

enum **lv\_file\_explorer\_file\_kind\_t**

*Values:*

enumerator **LV\_FILE\_EXPLORER\_FILE\_KIND\_DIR**

enumerator **LV\_FILE\_EXPLORER\_FILE\_KIND\_IMAGE**

enumerator **LV\_FILE\_EXPLORER\_FILE\_KIND\_AUDIO**

enumerator **LV\_FILE\_EXPLORER\_FILE\_KIND\_VIDEO**

enumerator **LV\_FILE\_EXPLORER\_FILE\_KIND\_FILE**

struct **\_lv\_file\_explorer\_t**

#### Public Members

lv\_obj\_t **obj**

lv\_obj\_t \***cont**

lv\_obj\_t \***head\_area**

lv\_obj\_t \***browser\_area**

lv\_obj\_t \***file\_table**

lv\_obj\_t \***path\_label**

lv\_obj\_t \***quick\_access\_area**

lv\_obj\_t \***list\_device**

lv\_obj\_t \***list\_places**

char \***home\_dir**

char \***music\_dir**

char \***pictures\_dir**

char \***video\_dir**

char \***docs\_dir**

char \***fs\_dir**

const char \***sel\_fn**

char **current\_path**[(128)]

*lv\_file\_explorer\_sort\_t* **sort**

struct **lv\_file\_explorer\_file\_table\_entry\_data\_t**

**Public Members**

*lv\_file\_explorer\_file\_kind\_t* **file\_kind**

**14.20.2 font\_manager**

**lv\_font\_manager.h**

**Functions**

**lv\_font\_manager\_t \*lv\_font\_manager\_create(uint32\_t recycle\_cache\_size)**

Create main font manager.

**Parametri**

**recycle\_cache\_size** -- number of fonts that were recently deleted from the cache.

**Ritorna**

pointer to main font manager.

**bool lv\_font\_manager\_delete(lv\_font\_manager\_t \*manager)**

Delete main font manager.

**Parametri**

**manager** -- pointer to main font manager.

**Ritorna**

return true if the deletion was successful.

**bool lv\_font\_manager\_add\_src(lv\_font\_manager\_t \*manager, const char \*name, const void \*src, const lv\_font\_class\_t \*class\_p)**

Add font resource.

**Parametri**

- **manager** -- pointer to main font manager.
- **name** -- font name.
- **src** -- font source. Need to strictly correspond to the font class.
- **class\_p** -- font class. eg. lv\_freetype\_font\_class, lv\_builtin\_font\_class.

**Ritorna**

return true if the add was successful.

**bool lv\_font\_manager\_add\_src\_static(lv\_font\_manager\_t \*manager, const char \*name, const void \*src, const lv\_font\_class\_t \*class\_p)**

Add font resource with static memory.

**Parametri**

- **manager** -- pointer to main font manager.
- **name** -- font name. It cannot be a local variable.
- **src** -- font source. Need to strictly correspond to the font class. And it cannot be a local variable.
- **class\_p** -- font class. E.g. lv\_freetype\_font\_class, lv\_builtin\_font\_class.

**Ritorna**

return true if the add was successful.

**bool lv\_font\_manager\_remove\_src(lv\_font\_manager\_t \*manager, const char \*name)**

Remove font resource.

**Parametri**

- **manager** -- pointer to main font manager.
- **name** -- font name.

**Ritorna**

return true if the remove was successful.

**lv\_font\_t \*lv\_font\_manager\_create\_font(lv\_font\_manager\_t \*manager, const char \*font\_family, uint32\_t render\_mode, uint32\_t size, uint32\_t style, *lv\_font\_kerning\_t* kerning)**

Create font.

**Parametri**

- **manager** -- pointer to main font manager.
- **font\_family** -- font family name. Matches the font resource name, using commas to separate different names. E.g. "my\_font\_1,my\_font\_2".
- **render\_mode** -- font render mode. see `lv_freetype_font_render_mode_t`.
- **size** -- font size in pixel.
- **style** -- font style. see `lv_freetype_font_style_t`.
- **kerning** -- kerning mode. see `lv_font_kerning_t`.

**Ritorna**

point to the created font.

`void lv_font_manager_delete_font(lv_font_manager_t *manager, lv_font_t *font)`

Delete font.

**Parametri**

- **manager** -- pointer to main font manager.
- **font** -- point to the font.

**Ritorna**

return true if the deletion was successful.

**lv\_font\_manager\_recycle.h****TypeDef**

`typedef struct _lv_font_manager_recycle_t lv_font_manager_recycle_t`

**Functions**

`lv_font_manager_recycle_t *lv_font_manager_recycle_create(uint32_t max_size)`

Create font recycle manager.

**Parametri**

- **max\_size** -- recycle size.

**Ritorna**

pointer to font recycle manager.

`void lv_font_manager_recycle_delete(lv_font_manager_recycle_t *manager)`

Delete font recycle manager.

**Parametri**

- **manager** -- pointer to font recycle manager.

`lv_font_t *lv_font_manager_recycle_get_reuse(lv_font_manager_recycle_t *manager, const lv_font_info_t *ft_info)`

Get a reusable font.

**Parametri**

- **manager** -- pointer to font recycle manager.
- **ft\_info** -- font info.

**Ritorna**

returns true on success.

`void lv_font_manager_recycle_set_reuse(lv_font_manager_recycle_t *manager, lv_font_t *font, const lv_font_info_t *ft_info)`

Set fonts to be reused.

**Parametri**

- **manager** -- pointer to font recycle manager.
- **ft\_info** -- font info.

### 14.20.3 fragment

#### lv\_fragment.h

Public header for Fragment

#### Typedef

typedef struct \_lv\_fragment\_manager\_t **lv\_fragment\_manager\_t**

#### Functions

**lv\_fragment\_manager\_t \*lv\_fragment\_manager\_create(lv\_fragment\_t \*parent)**

Create fragment manager instance

##### Parametri

**parent** -- Parent fragment if this manager is placed inside another fragment, can be null.

##### Ritorna

Fragment manager instance

**void lv\_fragment\_manager\_delete(lv\_fragment\_manager\_t \*manager)**

Destroy fragment manager instance

##### Parametri

**manager** -- Fragment manager instance

**void lv\_fragment\_manager\_create\_obj(lv\_fragment\_manager\_t \*manager)**

Create object of all fragments managed by this manager.

##### Parametri

**manager** -- Fragment manager instance

**void lv\_fragment\_manager\_delete\_obj(lv\_fragment\_manager\_t \*manager)**

Delete object created by all fragments managed by this manager. Instance of fragments will not be deleted.

##### Parametri

**manager** -- Fragment manager instance

**void lv\_fragment\_manager\_add(lv\_fragment\_manager\_t \*manager, lv\_fragment\_t \*fragment, lv\_obj\_t \*const \*container)**

Attach fragment to manager, and add to container.

##### Parametri

- **manager** -- Fragment manager instance
- **fragment** -- Fragment instance
- **container** -- Pointer to container object for manager to add objects to

**void lv\_fragment\_manager\_remove(lv\_fragment\_manager\_t \*manager, lv\_fragment\_t \*fragment)**

Detach and destroy fragment. If fragment is in navigation stack, remove from it.

##### Parametri

- **manager** -- Fragment manager instance
- **fragment** -- Fragment instance

**void lv\_fragment\_manager\_push(lv\_fragment\_manager\_t \*manager, lv\_fragment\_t \*fragment, lv\_obj\_t \*const \*container)**

Attach fragment to manager and add to navigation stack.

##### Parametri

- **manager** -- Fragment manager instance
- **fragment** -- Fragment instance
- **container** -- Pointer to container object for manager to add objects to

**bool lv\_fragment\_manager\_pop(lv\_fragment\_manager\_t \*manager)**

Remove the top-most fragment for stack

##### Parametri

**manager** -- Fragment manager instance

**Ritorna**

true if there is fragment to pop

```
void lv_fragment_manager_replace(lv_fragment_manager_t *manager, lv_fragment_t *fragment,  
                               lv_obj_t *const *container)
```

Replace fragment. Old item in the stack will be removed.

**Parametri**

- **manager** -- Fragment manager instance
- **fragment** -- Fragment instance
- **container** -- Pointer to container object for manager to add objects to

```
bool lv_fragment_manager_send_event(lv_fragment_manager_t *manager, int code, void *userdata)
```

Send event to top-most fragment

**Parametri**

- **manager** -- Fragment manager instance
- **code** -- User-defined ID of event
- **userdata** -- User-defined data

**Ritorna**

true if fragment returned true

```
size_t lv_fragment_manager_get_stack_size(lv_fragment_manager_t *manager)
```

Get stack size of this fragment manager

**Parametri**

- **manager** -- Fragment manager instance

**Ritorna**

Stack size of this fragment manager

```
lv_fragment_t *lv_fragment_manager_get_top(lv_fragment_manager_t *manager)
```

Get top most fragment instance

**Parametri**

- **manager** -- Fragment manager instance

**Ritorna**

Top most fragment instance

```
lv_fragment_t *lv_fragment_manager_find_by_container(lv_fragment_manager_t *manager, const  
                                                 lv_obj_t *container)
```

Find first fragment instance in the container

**Parametri**

- **manager** -- Fragment manager instance
- **container** -- Container which target fragment added to

**Ritorna**

First fragment instance in the container

```
lv_fragment_t *lv_fragment_manager_get_parent_fragment(lv_fragment_manager_t *manager)
```

Get parent fragment

**Parametri**

- **manager** -- Fragment manager instance

**Ritorna**

Parent fragment instance

```
lv_fragment_t *lv_fragment_create(const lv_fragment_class_t *cls, void *args)
```

Create a fragment instance.

**Parametri**

- **cls** -- Fragment class. This fragment must return non null object.
- **args** -- Arguments assigned by fragment manager

**Ritorna**

Fragment instance

```
void lv_fragment_delete(lv_fragment_t *fragment)
```

Destroy a fragment.

**Parametri**

**fragment** -- Fragment instance.

`lv_fragment_manager_t *lv_fragment_get_manager(lv_fragment_t *fragment)`

Get associated manager of this fragment

**Parametri**

**fragment** -- Fragment instance

**Ritorna**

Fragment manager instance

`lv_obj_t *const *lv_fragment_get_container(lv_fragment_t *fragment)`

Get container object of this fragment

**Parametri**

**fragment** -- Fragment instance

**Ritorna**

Reference to container object

`lv_fragment_t *lv_fragment_get_parent(lv_fragment_t *fragment)`

Get parent fragment of this fragment

**Parametri**

**fragment** -- Fragment instance

**Ritorna**

Parent fragment

`lv_obj_t *lv_fragment_create_obj(lv_fragment_t *fragment, lv_obj_t *container)`

Create object by fragment.

**Parametri**

- **fragment** -- Fragment instance.
- **container** -- Container of the objects should be created upon.

**Ritorna**

Created object

`void lv_fragment_delete_obj(lv_fragment_t *fragment)`

Delete created object of a fragment

**Parametri**

**fragment** -- Fragment instance.

`void lv_fragment_recreate_obj(lv_fragment_t *fragment)`

Destroy obj in fragment, and recreate them.

**Parametri**

**fragment** -- Fragment instance

`struct _lv_fragment_t`

**Public Members**

`const lv_fragment_class_t *cls`

Class of this fragment

`lv_fragment_managed_states_t *managed`

Managed fragment states. If not null, then this fragment is managed.

**Avvertimento**

Don't modify values inside this struct!

`lv_fragment_manager_t *child_manager`

Child fragment manager

`lv_obj_t *obj`

lv\_obj returned by create\_obj\_cb

struct `_lv_fragment_class_t`

### Public Members

`void (*constructor_cb)(lv_fragment_t *self, void *args)`

Constructor function for fragment class

**Param self**

Fragment instance

**Param args**

Arguments assigned by fragment manager

`void (*destructor_cb)(lv_fragment_t *self)`

Destructor function for fragment class

**Param self**

Fragment instance, will be freed after this call

`void (*attached_cb)(lv_fragment_t *self)`

Fragment attached to manager

**Param self**

Fragment instance

`void (*detached_cb)(lv_fragment_t *self)`

Fragment detached from manager

**Param self**

Fragment instance

`lv_obj_t *(*create_obj_cb)(lv_fragment_t *self, lv_obj_t *container)`

Create objects

**Param self**

Fragment instance

**Param container**

Container of the objects should be created upon

**Return**

Created object, NULL if multiple objects has been created

`void (*obj_created_cb)(lv_fragment_t *self, lv_obj_t *obj)`

**Param self**

Fragment instance

**Param obj**

lv\_obj returned by create\_obj\_cb

`void (*obj_will_delete_cb)(lv_fragment_t *self, lv_obj_t *obj)`

Called before objects in the fragment will be deleted.

**Param self**

Fragment instance

**Param obj**

object with this fragment

void (\***obj\_deleted\_cb**)(lv\_fragment\_t \*self, lv\_obj\_t \*obj)

Called when the object created by fragment received LV\_EVENT\_DELETE event

**Param self**

Fragment instance

**Param obj**

object with this fragment

bool (\***event\_cb**)(lv\_fragment\_t \*self, int code, void \*userdata)

Handle event

**Param self**

Fragment instance

**Param which**

User-defined ID of event

**Param data1**

User-defined data

**Param data2**

User-defined data

**size\_t instance\_size**

REQUIRED: Allocation size of fragment

**lv\_fragment\_private.h**struct **\_lv\_fragment\_managed\_states\_t**

#include &lt;lv\_fragment\_private.h&gt; Fragment states

**Public Members**const lv\_fragment\_class\_t \***cls**

Class of the fragment

lv\_fragment\_manager\_t \***manager**

Manager the fragment attached to

lv\_obj\_t \*const \***container**

Container object the fragment adding view to

lv\_fragment\_t \***instance**

Fragment instance

bool **obj\_created**

true between create\_obj\_cb and obj\_deleted\_cb

bool **destroying\_obj**

true before lv\_fragment\_delete\_obj is called. Don't touch any object if this is true

bool **in\_stack**

true if this fragment is in navigation stack that can be popped

## 14.20.4 gridnav

[lv\\_gridnav.h](#)

### Enums

enum **lv\_gridnav\_ctrl\_t**

*Values:*

enumerator **LV\_GRIDNAV\_CTRL\_NONE**

enumerator **LV\_GRIDNAV\_CTRL\_ROLLOVER**

If there is no next/previous object in a direction, the focus goes to the object in the next/previous row (on left/right keys) or first/last row (on up/down keys)

enumerator **LV\_GRIDNAV\_CTRL\_SCROLL\_FIRST**

If an arrow is pressed and the focused object can be scrolled in that direction then it will be scrolled instead of going to the next/previous object. If there is no more room for scrolling the next/previous object will be focused normally

enumerator **LV\_GRIDNAV\_CTRL\_HORIZONTAL\_MOVE\_ONLY**

Only use left/right keys for grid navigation. Up/down key events will be sent to the focused object.

enumerator **LV\_GRIDNAV\_CTRL\_VERTICAL\_MOVE\_ONLY**

Only use up/down keys for grid navigation. Left/right key events will be sent to the focused object.

### Functions

void **lv\_gridnav\_add**(lv\_obj\_t \*obj, [lv\\_gridnav\\_ctrl\\_t](#) ctrl)

Add grid navigation feature to an object. It expects the children to be arranged into a grid-like layout. Although it's not required to have pixel perfect alignment. This feature makes possible to use keys to navigate among the children and focus them. The keys other than arrows and press/release related events are forwarded to the focused child.

#### Parametri

- **obj** -- pointer to an object on which navigation should be applied.
- **ctrl** -- control flags from [lv\\_gridnav\\_ctrl\\_t](#).

void **lv\_gridnav\_remove**(lv\_obj\_t \*obj)

Remove the grid navigation support from an object

#### Parametri

- **obj** -- pointer to an object

void **lv\_gridnav\_set\_focused**(lv\_obj\_t \*cont, lv\_obj\_t \*to\_focus, [lv\\_anim\\_enable\\_t](#) anim\_en)

Manually focus an object on gridnav container

#### Parametri

- **cont** -- pointer to a gridnav container
- **to\_focus** -- pointer to an object to focus
- **anim\_en** -- LV\_ANIM\_ON/OFF

## 14.20.5 ime

[lv\\_ime\\_pinyin.h](#)

### Defines

**LVIME\_PINYIN\_K9\_MAX\_INPUT**

## Enums

enum **lv\_ime\_pinyin\_mode\_t**

*Values:*

enumerator **LVIME\_PINYIN\_MODE\_K26**

enumerator **LVIME\_PINYIN\_MODE\_K9**

enumerator **LVIME\_PINYIN\_MODE\_K9\_NUMBER**

## Functions

lv\_obj\_t \***lv\_ime\_pinyin\_create**(lv\_obj\_t \*parent)

void **lv\_ime\_pinyin\_set\_keyboard**(lv\_obj\_t \*obj, lv\_obj\_t \*kb)

Set the keyboard of Pinyin input method.

**Parametri**

- **obj** -- pointer to a Pinyin input method object
- **kb** -- pointer to a Pinyin input method keyboard

void **lv\_ime\_pinyin\_set\_dict**(lv\_obj\_t \*obj, *lv\_pinyin\_dict\_t* \*dict)

Set the dictionary of Pinyin input method.

**Parametri**

- **obj** -- pointer to a Pinyin input method object
- **dict** -- pointer to a Pinyin input method dictionary

void **lv\_ime\_pinyin\_set\_mode**(lv\_obj\_t \*obj, *lv\_ime\_pinyin\_mode\_t* mode)

Set mode, 26-key *input(k26)* or 9-key *input(k9)*.

**Parametri**

- **obj** -- pointer to a Pinyin input method object
- **mode** -- the mode from 'lv\_ime\_pinyin\_mode\_t'

lv\_obj\_t \***lv\_ime\_pinyin\_get\_kb**(lv\_obj\_t \*obj)

Set the dictionary of Pinyin input method.

**Parametri**

**obj** -- pointer to a Pinyin IME object

**Ritorna**

pointer to the Pinyin IME keyboard

lv\_obj\_t \***lv\_ime\_pinyin\_get\_cand\_panel**(lv\_obj\_t \*obj)

Set the dictionary of Pinyin input method.

**Parametri**

**obj** -- pointer to a Pinyin input method object

**Ritorna**

pointer to the Pinyin input method candidate panel

const *lv\_pinyin\_dict\_t* \***lv\_ime\_pinyin\_get\_dict**(lv\_obj\_t \*obj)

Set the dictionary of Pinyin input method.

**Parametri**

**obj** -- pointer to a Pinyin input method object

**Ritorna**

pointer to the Pinyin input method dictionary

## Variables

const lv\_obj\_class\_t **lv\_ime\_pinyin\_class**

struct **lv\_pinyin\_dict\_t**

### Public Members

const char \*const **py**

const char \*const **py\_mb**

struct **ime\_pinyin\_k9\_py\_str\_t**

### Public Members

char **py\_str[7]**

**lv\_ime\_pinyin\_private.h**

struct **\_lv\_ime\_pinyin\_t**

### Public Members

lv\_obj\_t **obj**

lv\_obj\_t \***kb**

lv\_obj\_t \***cand\_panel**

const *lv\_pinyin\_dict\_t* \***dict**

*lv\_ll\_t* **k9\_legal\_py\_ll**

char \***cand\_str**

char **input\_char[16]**

char **k9\_input\_str[7 + 1]**

uint16\_t **k9\_py\_ll\_pos**

uint16\_t **k9\_legal\_py\_count**

uint16\_t **k9\_input\_str\_len**

uint16\_t **ta\_count**

```
uint16_t cand_num
```

```
uint16_t py_page
```

```
uint16_t py_num[26]
```

```
uint16_t py_pos[26]
```

```
lv_ime_pinyin_mode_t mode
```

## 14.20.6 imgfont

**lv\_imgfont.h**

**Typedef**

```
typedef const void *(*lv_imgfont_get_path_cb_t)(const lv_font_t *font, uint32_t unicode, uint32_t
unicode_next, int32_t *offset_y, void *user_data)
```

**Functions**

**lv\_font\_t \*lv\_imgfont\_create**(uint16\_t height, *lv\_imgfont\_get\_path\_cb\_t* path\_cb, void \*user\_data)

Creates a image font with info parameter specified.

**Parametri**

- **height** -- font size
- **path\_cb** -- a function to get the image path name of character.
- **user\_data** -- pointer to user data

**Ritorna**

pointer to the new imgfont or NULL if create error.

**void lv\_imgfont\_destroy(lv\_font\_t \*font)**

Destroy a image font that has been created.

**Parametri**

**font** -- pointer to image font handle.

## 14.20.7 monkey

**lv\_monkey.h**

**Typedef**

```
typedef struct _lv_monkey_t lv_monkey_t
```

**Functions**

**void lv\_monkey\_config\_init(lv\_monkey\_config\_t \*config)**

Initialize a monkey config with default values

**Parametri**

**config** -- pointer to 'lv\_monkey\_config\_t' variable to initialize

**lv\_monkey\_t \*lv\_monkey\_create(const lv\_monkey\_config\_t \*config)**

Create monkey for test

**Parametri**

**config** -- pointer to 'lv\_monkey\_config\_t' variable

**Ritorna**  
pointer to the created monkey

**lv\_indev\_t \*lv\_monkey\_get\_indev(*lv\_monkey\_t* \*monkey)**

Get monkey input device

**Parametri**

- **monkey** -- pointer to a monkey

**Ritorna**  
pointer to the input device

**void lv\_monkey\_set\_enable(*lv\_monkey\_t* \*monkey, bool en)**

Enable monkey

**Parametri**

- **monkey** -- pointer to a monkey
- **en** -- set to true to enable

**bool lv\_monkey\_get\_enable(*lv\_monkey\_t* \*monkey)**

Get whether monkey is enabled

**Parametri**

- **monkey** -- pointer to a monkey

**Ritorna**  
return true if monkey enabled

**void lv\_monkey\_set\_user\_data(*lv\_monkey\_t* \*monkey, void \*user\_data)**

Set the user\_data field of the monkey

**Parametri**

- **monkey** -- pointer to a monkey
- **user\_data** -- pointer to the new user\_data.

**void \*lv\_monkey\_get\_user\_data(*lv\_monkey\_t* \*monkey)**

Get the user\_data field of the monkey

**Parametri**

- **monkey** -- pointer to a monkey

**Ritorna**  
the pointer to the user\_data of the monkey

**void lv\_monkey\_delete(*lv\_monkey\_t* \*monkey)**

Delete monkey

**Parametri**

- **monkey** -- pointer to monkey

struct **lv\_range\_t**

#### Public Members

int32\_t **min**

int32\_t **max**

struct **lv\_urange\_t**

#### Public Members

uint32\_t **min**

uint32\_t **max**

---

struct **\_lv\_monkey\_config\_t**

#### Public Members

*lv\_indev\_type\_t* **type**

Input device type

*lv\_urange\_t* **period\_range**

Monkey execution period

*lv\_range\_t* **input\_range**

The range of input value

**lv\_monkey\_private.h**

### 14.20.8 observer

**lv\_observer.h**

#### Typedef

typedef void (\***lv\_observer\_cb\_t**)(lv\_observer\_t \*observer, *lv\_subject\_t* \*subject)

Callback called to notify Observer that Subject's value has changed

**Param observer**

pointer to Observer

**Param subject**

pointer to Subject being observed

#### Enums

enum **lv\_subject\_type\_t**

Values for *lv\_subject\_t*'s **type** field

*Values:*

enumerator **LV SUBJECT TYPE INVALID**

indicates Subject not initialized yet

enumerator **LV SUBJECT TYPE NONE**

a null value like None or NILt

enumerator **LV SUBJECT TYPE INT**

an int32\_t

enumerator **LV SUBJECT TYPE FLOAT**

a float, requires LV USE FLOAT 1

enumerator **LV SUBJECT TYPE POINTER**

a void pointer

enumerator **LV SUBJECT TYPE COLOR**

an *lv\_color\_t*

enumerator **LV SUBJECT TYPE GROUP**

an array of Subjects

enumerator **LV SUBJECT TYPE STRING**

a char pointer

## Functions

**void lv\_subject\_init\_int(*lv\_subject\_t* \*subject, int32\_t value)**

Initialize an integer-type Subject.

**Parametri**

- **subject** -- pointer to Subject
- **value** -- initial value

**void lv\_subject\_set\_int(*lv\_subject\_t* \*subject, int32\_t value)**

Set value of an integer Subject and notify Observers.

**Parametri**

- **subject** -- pointer to Subject
- **value** -- new value

**int32\_t lv\_subject\_get\_int(*lv\_subject\_t* \*subject)**

Get current value of an integer Subject.

**Parametri**

**subject** -- pointer to Subject

**Ritorna**

current value

**int32\_t lv\_subject\_get\_previous\_int(*lv\_subject\_t* \*subject)**

Get previous value of an integer Subject.

**Parametri**

**subject** -- pointer to Subject

**Ritorna**

current value

**void lv\_subject\_init\_float(*lv\_subject\_t* \*subject, float value)**

Initialize an float-type Subject.

**Parametri**

- **subject** -- pointer to Subject
- **value** -- initial value

**void lv\_subject\_set\_float(*lv\_subject\_t* \*subject, float value)**

Set value of an float Subject and notify Observers.

**Parametri**

- **subject** -- pointer to Subject
- **value** -- new value

**float lv\_subject\_get\_float(*lv\_subject\_t* \*subject)**

Get current value of an float Subject.

**Parametri**

**subject** -- pointer to Subject

**Ritorna**

current value

**float lv\_subject\_get\_previous\_float(*lv\_subject\_t* \*subject)**

Get previous value of an float Subject.

**Parametri**

**subject** -- pointer to Subject

**Ritorna**

current value

```
void lv_subject_init_string(lv_subject_t *subject, char *buf, char *prev_buf, size_t size, const char *value)
```

Initialize a string-type Subject.

#### Nota

A string Subject stores its own copy of the string, not just the pointer.

#### Parametri

- **subject** -- pointer to Subject
- **buf** -- pointer to buffer to store string
- **prev\_buf** -- pointer to buffer to store previous string; can be NULL if not used
- **size** -- size of buffer(s)
- **value** -- initial value of string, e.g. "hello"

```
void lv_subject_copy_string(lv_subject_t *subject, const char *buf)
```

Copy a string to a Subject and notify Observers if it changed.

#### Parametri

- **subject** -- pointer to Subject
- **buf** -- new string

```
void lv_subject_snprintf(lv_subject_t *subject, const char *format, ...)
```

Format a new string, updating Subject, and notify Observers if it changed.

#### Parametri

- **subject** -- pointer to Subject
- **format** -- format string

```
const char *lv_subject_get_string(lv_subject_t *subject)
```

Get current value of a string Subject.

#### Parametri

**subject** -- pointer to Subject

#### Ritorna

pointer to buffer containing current value

```
const char *lv_subject_get_previous_string(lv_subject_t *subject)
```

Get previous value of a string Subject.

#### Nota

NULL will be returned if NULL was passed in `lv_subject_init_string()` as `prev_buf`.

#### Parametri

**subject** -- pointer to Subject

#### Ritorna

pointer to buffer containing previous value

```
void lv_subject_init_pointer(lv_subject_t *subject, void *value)
```

Initialize a pointer-type Subject.

#### Parametri

- **subject** -- pointer to Subject
- **value** -- initial value

```
void lv_subject_set_pointer(lv_subject_t *subject, void *ptr)
```

Set value of a pointer Subject and notify Observers (regardless of whether it changed).

#### Parametri

- **subject** -- pointer to Subject
- **ptr** -- new value

`const void *lv_subject_get_pointer(lv_subject_t *subject)`

Get current value of a pointer Subject.

**Parametri**

**subject** -- pointer to Subject

**Ritorna**

current value

`const void *lv_subject_get_previous_pointer(lv_subject_t *subject)`

Get previous value of a pointer Subject.

**Parametri**

**subject** -- pointer to Subject

**Ritorna**

previous value

`void lv_subject_init_color(lv_subject_t *subject, lv_color_t color)`

Initialize a color-type Subject.

**Parametri**

- **subject** -- pointer to Subject
- **color** -- initial value

`void lv_subject_set_color(lv_subject_t *subject, lv_color_t color)`

Set value of a color Subject and notify Observers if it changed.

**Parametri**

- **subject** -- pointer to Subject
- **color** -- new value

`lv_color_t lv_subject_get_color(lv_subject_t *subject)`

Get current value of a color Subject.

**Parametri**

**subject** -- pointer to Subject

**Ritorna**

current value

`lv_color_t lv_subject_get_previous_color(lv_subject_t *subject)`

Get previous value of a color Subject.

**Parametri**

**subject** -- pointer to Subject

**Ritorna**

previous value

`void lv_subject_init_group(lv_subject_t *group_subject, lv_subject_t *list[], uint32_t list_len)`

Initialize a Group-type Subject.

**Parametri**

- **group\_subject** -- pointer to Group-type Subject
- **list** -- list of other Subject addresses; when any of these have values updated, Observers of **group\_subject** will be notified.
- **list\_len** -- number of elements in **list[]**

`void lv_subject_deinit(lv_subject_t *subject)`

Remove all Observers from a Subject and free allocated memory, and delete any associated Widget-Binding events. This leaves **subject** "disconnected" from all Observers and all associated Widget events established through Widget Binding.

**Nota**

This can safely be called regardless of whether any Observers added with `lv_subject_add_observer_obj()` or bound to a Widget Property with one of the `..._bind_...` functions.

**Parametri**

- **subject** -- pointer to Subject

`lv_subject_t *lv_subject_get_group_element(lv_subject_t *subject, int32_t index)`

Get an element from Subject Group's list.

**Parametri**

- **subject** -- pointer to Group-type Subject
- **index** -- index of element to get

**Ritorna**

pointer to indexed Subject from list, or NULL if index is out of bounds

`lv_observer_t *lv_subject_add_observer(lv_subject_t *subject, lv_observer_cb_t observer_cb, void *user_data)`

Add Observer to Subject. When Subject's value changes `observer_cb` will be called.

**Parametri**

- **subject** -- pointer to Subject
- **observer\_cb** -- notification callback
- **user\_data** -- optional user data

**Ritorna**

pointer to newly-created Observer

`lv_observer_t *lv_subject_add_observer_obj(lv_subject_t *subject, lv_observer_cb_t observer_cb, lv_obj_t *obj, void *user_data)`

Add Observer to Subject for a Widget. When the Widget is deleted, Observer will be unsubscribed from Subject automatically.

 **Nota**

Do not call `lv_observer_remove()` on Observers created this way. Only clean up such Observers by either:

- deleting the Widget, or
- calling `lv_subject_deinit()` to gracefully de-couple and remove all Observers.

**Parametri**

- **subject** -- pointer to Subject
- **observer\_cb** -- notification callback
- **obj** -- pointer to Widget
- **user\_data** -- optional user data

**Ritorna**

pointer to newly-created Observer

`lv_observer_t *lv_subject_add_observer_with_target(lv_subject_t *subject, lv_observer_cb_t observer_cb, void *target, void *user_data)`

Add an Observer to a Subject and also save a target pointer.

**Parametri**

- **subject** -- pointer to Subject
- **observer\_cb** -- notification callback
- **target** -- any pointer (NULL is okay)
- **user\_data** -- optional user data

**Ritorna**

pointer to newly-created Observer

`void lv_observer_remove(lv_observer_t *observer)`

Remove Observer from its Subject.

**Parametri**

**observer** -- pointer to Observer

`void lv_obj_remove_from_subject(lv_obj_t *obj, lv_subject_t *subject)`

Remove Observers associated with Widget **obj** from specified **subject** or all Subjects.

### Nota

This function can be used e.g. when a Widget's Subject(s) needs to be replaced by other Subject(s)

#### Parametri

- **obj** -- pointer to Widget whose Observers should be removed
- **subject** -- Subject to remove Widget from, or NULL to remove from all Subjects

`void *lv_observer_get_target(lv_observer_t *observer)`

Get target of an Observer.

#### Parametri

- **observer** -- pointer to Observer

#### Ritorna

pointer to saved target

`lv_obj_t *lv_observer_get_target_obj(lv_observer_t *observer)`

Get target Widget of Observer. This is the same as `lv_observer_get_target()`, except it returns **target** as an `lv_obj_t *`.

#### Parametri

- **observer** -- pointer to Observer

#### Ritorna

pointer to saved Widget target

`void *lv_observer_get_user_data(const lv_observer_t *observer)`

Get Observer's user data.

#### Parametri

- **observer** -- pointer to Observer

#### Ritorna

void pointer to saved user data

`void lv_subject_notify(lv_subject_t *subject)`

Notify all Observers of Subject.

#### Parametri

- **subject** -- pointer to Subject

`void lv_obj_add_subject_increment_event(lv_obj_t *obj, lv_subject_t *subject, lv_event_code_t trigger, int32_t step, int32_t min, int32_t max)`

Add an event handler to increment (or decrement) the value of a subject on a trigger.

#### Parametri

- **obj** -- pointer to a widget
- **subject** -- pointer to a subject to change
- **trigger** -- the trigger on which the subject should be changed
- **step** -- value to add on trigger
- **min** -- the minimum value
- **max** -- the maximum value

`void lv_obj_add_subject_set_int_event(lv_obj_t *obj, lv_subject_t *subject, lv_event_code_t trigger, int32_t value)`

Set the value of an integer subject.

#### Parametri

- **obj** -- pointer to a widget
- **subject** -- pointer to a subject to change
- **trigger** -- the trigger on which the subject should be changed
- **value** -- the value to set

```
void lv_obj_add_subject_set_float_event(lv_obj_t *obj, lv_subject_t *subject, lv_event_code_t trigger, float value)
```

Set the value of a float subject.

**Parametri**

- **obj** -- pointer to a widget
- **subject** -- pointer to a subject to change
- **trigger** -- the trigger on which the subject should be changed
- **value** -- the value to set

```
void lv_obj_add_subject_set_string_event(lv_obj_t *obj, lv_subject_t *subject, lv_event_code_t trigger, const char *value)
```

Set the value of a string subject.

**Parametri**

- **obj** -- pointer to a widget
- **subject** -- pointer to a subject to change
- **trigger** -- the trigger on which the subject should be changed
- **value** -- the value to set

```
lv_observer_t *lv_obj_bind_style(lv_obj_t *obj, const lv_style_t *style, lv_style_selector_t selector, lv_subject_t *subject, int32_t ref_value)
```

Disable a style if a subject's value is not equal to a reference value

**Parametri**

- **obj** -- pointer to Widget
- **style** -- pointer to a style
- **selector** -- pointer to a selector
- **subject** -- pointer to Subject
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_eq(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is equal to a reference value, clear flag otherwise.

**Parametri**

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_OBJ\_FLAG\_HID-DEN)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_not_eq(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is not equal to a reference value, clear flag otherwise.

**Parametri**

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_OBJ\_FLAG\_HID-DEN)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_gt(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is greater than a reference value, clear flag otherwise.

**Parametri**

- **obj** -- pointer to Widget

- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_OBJ\_FLAG\_HID-DEN)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_ge(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is greater than or equal to a reference value, clear flag otherwise.

**Parametri**

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_OBJ\_FLAG\_HID-DEN)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_lt(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is less than a reference value, clear flag otherwise.

**Parametri**

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_OBJ\_FLAG\_HID-DEN)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_flag_if_le(lv_obj_t *obj, lv_subject_t *subject, lv_obj_flag_t flag, int32_t ref_value)
```

Set Widget's flag(s) if an integer Subject's value is less than or equal to a reference value, clear flag otherwise.

**Parametri**

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **flag** -- flag(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_OBJ\_FLAG\_HID-DEN)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_state_if_eq(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)
```

Set Widget's state(s) if an integer Subject's value is equal to a reference value, clear flag otherwise.

**Parametri**

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_STATE\_CHECKED)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_state_if_not_eq(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)
```

Set a Widget's state(s) if an integer Subject's value is not equal to a reference value, clear flag otherwise

**Parametri**

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_STATE\_CHECKED)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_state_if_gt(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)
```

Set Widget's state(s) if an integer Subject's value is greater than a reference value, clear flag otherwise.

**Parametri**

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_STATE\_CHECKED)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_state_if_ge(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)
```

Set Widget's state(s) if an integer Subject's value is greater than or equal to a reference value, clear flag otherwise.

**Parametri**

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_STATE\_CHECKED)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_state_if_lt(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)
```

Set Widget's state(s) if an integer Subject's value is less than a reference value, clear flag otherwise.

**Parametri**

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_STATE\_CHECKED)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_state_if_le(lv_obj_t *obj, lv_subject_t *subject, lv_state_t state, int32_t ref_value)
```

Set Widget's state(s) if an integer Subject's value is less than or equal to a reference value, clear flag otherwise.

**Parametri**

- **obj** -- pointer to Widget
- **subject** -- pointer to Subject
- **state** -- state(s) (can be bit-wise OR-ed) to set or clear (e.g. LV\_STATE\_CHECKED)
- **ref\_value** -- reference value to compare Subject's value with

**Ritorna**

pointer to newly-created Observer

```
lv_observer_t *lv_obj_bind_checked(lv_obj_t *obj, lv_subject_t *subject)
```

Set an integer Subject to 1 when a Widget is checked and set it 0 when unchecked, and clear Widget's checked state when Subject's value changes to 0 and set it when non-zero.

#### Nota

Ensure Widget's `LV_OBJ_FLAG_CHECKABLE` flag is set.

#### Parametri

- **obj** -- pointer to Widget
- **subject** -- pointer to a Subject

#### Ritorna

pointer to newly-created Observer

`lv_observer_t *lv_label_bind_text(lv_obj_t *obj, lv_subject_t *subject, const char *fmt)`

Bind an integer, string, or pointer Subject to a Label.

#### Nota

`fmt == NULL` can be used only with string and pointer Subjects.

#### Nota

If Subject is a pointer and `fmt == NULL`, pointer must point to a \0 terminated string.

#### Parametri

- **obj** -- pointer to Label
- **subject** -- pointer to Subject
- **fmt** -- optional printf-like format string with 1 format specifier (e.g. "%d °C") or NULL to bind to the value directly.

#### Ritorna

pointer to newly-created Observer

`lv_observer_t *lv_arc_bind_value(lv_obj_t *obj, lv_subject_t *subject)`

Bind an integer subject to an Arc's value.

#### Parametri

- **obj** -- pointer to Arc
- **subject** -- pointer to Subject

#### Ritorna

pointer to newly-created Observer

`lv_observer_t *lv_slider_bind_value(lv_obj_t *obj, lv_subject_t *subject)`

Bind an integer Subject to a Slider's value.

#### Parametri

- **obj** -- pointer to Slider
- **subject** -- pointer to Subject

#### Ritorna

pointer to newly-created Observer

`lv_observer_t *lv_roller_bind_value(lv_obj_t *obj, lv_subject_t *subject)`

Bind an integer Subject to a Roller's value.

#### Parametri

- **obj** -- pointer to Roller
- **subject** -- pointer to Subject

#### Ritorna

pointer to newly-created Observer

`lv_observer_t *lv_dropdown_bind_value(lv_obj_t *obj, lv_subject_t *subject)`

Bind an integer Subject to a Dropdown's value.

#### Parametri

- **obj** -- pointer to Dropdown
- **subject** -- pointer to Subject

#### Ritorna

pointer to newly-created Observer

union **lv\_subject\_value\_t**

#include <lv\_observer.h> A common type to handle all the various observable types in the same way

### Public Members

`int32_t num`

Integer number (opacity, enums, booleans or "normal" numbers)

`const void *pointer`

Constant pointer (string buffer, format string, font, cone text, etc.)

`lv_color_t color`

Color

`float float_v`

Floating point value

struct **lv\_subject\_t**

#include <lv\_observer.h> The Subject (an observable value)

### Public Members

`lv_ll_t subs_ll`

Subscribers

`lv_subject_value_t value`

Current value

`lv_subject_value_t prev_value`

Previous value

`void *user_data`

Additional parameter, can be used freely by user

`uint32_t type`

One of the LV SUBJECT\_TYPE\_... values

`uint32_t size`

String buffer size or group length

`uint32_t notify_restart_query`

If an Observer was deleted during notification, start notifying from the beginning.

**lv\_observer\_private.h****struct \_lv\_observer\_t**

`#include <lv_observer_private.h>` The observer object: a descriptor returned when subscribing LVGL widgets to subjects

**Public Members*****lv\_subject\_t* \*subject**

Observed subject

***lv\_observer\_cb\_t* cb**

Callback that notifies when value changes

**void \*target**

A target for the observer, e.g. a widget or any pointer

**void \*user\_data**

Additional parameter supplied when subscribing

**uint32\_t auto\_free\_user\_data**

Automatically free user data when observer is removed

**uint32\_t notified**

Was observer already notified?

**uint32\_t for\_obj**

Is target a pointer to a Widget (`lv_obj_t *`)?

**14.20.9 snapshot****lv\_snapshot.h****Functions****lv\_draw\_buf\_t \*lv\_snapshot\_take(lv\_obj\_t \*obj, lv\_color\_format\_t cf)**

Take snapshot for object with its children, create the draw buffer as needed.

**Parametri**

- **obj** -- the object to generate snapshot.
- **cf** -- color format for generated image.

**Ritorna**

a pointer to an draw buffer containing snapshot image, or NULL if failed.

**lv\_draw\_buf\_t \*lv\_snapshot\_create\_draw\_buf(lv\_obj\_t \*obj, lv\_color\_format\_t cf)**

Create a draw buffer to store the snapshot image for object.

**Parametri**

- **obj** -- the object to generate snapshot.
- **cf** -- color format for generated image.

**Ritorna**

a pointer to an draw buffer ready for taking snapshot, or NULL if failed.

**lv\_result\_t lv\_snapshot\_reshape\_draw\_buf(lv\_obj\_t \*obj, lv\_draw\_buf\_t \*draw\_buf)**

Reshape the draw buffer to prepare for taking snapshot for obj. This is usually used to check if the existing draw buffer is enough for obj snapshot. If return LV\_RESULT\_INVALID, you should create a new one.

**Parametri**

- **draw\_buf** -- the draw buffer to reshape.
- **obj** -- the object to generate snapshot.

```
lv_result_t lv_snapshot_take_to_draw_buf(lv_obj_t *obj, lv_color_format_t cf, lv_draw_buf_t
                                         *draw_buf)
```

Take snapshot for object with its children, save image info to provided buffer.

#### Parametri

- **obj** -- the object to generate snapshot.
- **cf** -- color format for new snapshot image. It could differ with cf of **draw\_buf** as long as the new cf will fit in.
- **draw\_buf** -- the draw buffer to store the image result. It's reshaped automatically.

#### Ritorna

LV\_RESULT\_OK on success, LV\_RESULT\_INVALID on error.

```
void lv_snapshot_free(lv_image_dsc_t *dsc)
```

*Deprecated:*

Use **lv\_draw\_buf\_destroy** instead.

Free the snapshot image returned by **lv\_snapshot\_take**

#### Parametri

- **dsc** -- the image descriptor generated by **lv\_snapshot\_take**.

```
lv_result_t lv_snapshot_take_to_buf(lv_obj_t *obj, lv_color_format_t cf, lv_image_dsc_t *dsc, void
                                    *buf, uint32_t buf_size)
```

Take snapshot for object with its children, save image info to provided buffer.

*Deprecated:*

Use **lv\_snapshot\_take\_to\_draw\_buf** instead.

#### Parametri

- **obj** -- the object to generate snapshot.
- **cf** -- color format for generated image.
- **dsc** -- image descriptor to store the image result.
- **buf** -- the buffer to store image data. It must meet align requirement.
- **buf\_size** -- provided buffer size in bytes.

#### Ritorna

LV\_RESULT\_OK on success, LV\_RESULT\_INVALID on error.

## 14.20.10 sysmon

### lv\_sysmon.h

#### Functions

```
lv_obj_t *lv_sysmon_create(lv_display_t *disp)
```

Create a new system monitor label

#### Parametri

- **disp** -- create the sys. mon. on this display's system layer

#### Ritorna

the create label

```
void lv_sysmon_show_performance(lv_display_t *disp)
```

Show system performance monitor: CPU usage and FPS count

#### Parametri

- **disp** -- target display, NULL: use the default displays

```
void lv_sysmon_hide_performance(lv_display_t *disp)
```

Hide system performance monitor

#### Parametri

- **disp** -- target display, NULL: use the default

**void lv\_sysmon\_performance\_dump(lv\_display\_t \*disp)**  
 Dump the FPS data recorded between the last and current dump call.

**Parametri**

**disp** -- target display, NULL: use the default

**void lv\_sysmon\_performance\_resume(lv\_display\_t \*disp)**

Resume the system performance monitor.

**Parametri**

**disp** -- target display, NULL: use the default

**void lv\_sysmon\_performance\_pause(lv\_display\_t \*disp)**

Pause the system performance monitor.

**Nota**

When the sysmon is stopped you can use `lv_sysmon_dump_performance` to get performance information. See `lv_sysmon_dump_performance` for more information.

**Parametri**

**disp** -- target display, NULL: use the default

**void lv\_sysmon\_show\_memory(lv\_display\_t \*disp)**

Show system memory monitor: used memory and the memory fragmentation

**Parametri**

**disp** -- target display, NULL: use the default displays

**void lv\_sysmon\_hide\_memory(lv\_display\_t \*disp)**

Hide system memory monitor

**Parametri**

**disp** -- target display, NULL: use the default displays

**lv\_sysmon\_private.h****Functions**

**void lv\_sysmon\_builtin\_init(void)**

Initialize built-in system monitor, such as performance and memory monitor.

**void lv\_sysmon\_builtin\_deinit(void)**

DeInitialize built-in system monitor, such as performance and memory monitor.

**struct \_lv\_sysmon\_backend\_data\_t****Public Members**

*lv\_subject\_t* **subject**

*lv\_timer\_t* \***timer**

**struct \_lv\_sysmon\_perf\_info\_t****Public Members**

**bool init**

```
uint32_t refr_start
uint32_t refr_interval_sum
uint32_t refr_elaps_sum
uint32_t refr_cnt
uint32_t render_start
uint32_t render_elaps_sum
uint32_t render_cnt
uint32_t flush_in_render_start
uint32_t flush_in_render_elaps_sum
uint32_t flush_not_in_render_start
uint32_t flush_not_in_render_elaps_sum
uint32_t last_report_timestamp
uint32_t render_in_progress
struct _lv_sysmon_perf_info_t measured
uint32_t fps
uint32_t cpu
uint32_t refr_avg_time
uint32_t render_avg_time
    Pure rendering time without flush time
uint32_t flush_avg_time
    Pure flushing time without rendering time
uint32_t cpu_avg_total
uint32_t fps_avg_total
uint32_t run_cnt
struct _lv_sysmon_perf_info_t calculated
```

## 14.20.11 test

**lv\_test.h**

**lv\_test\_display.h**

### Functions

`lv_display_t *lv_test_display_create(int32_t hor_res, int32_t ver_res)`

**lv\_test\_helpers.h**

### Defines

`LV_HEAP_CHECK(x)`

`LV_TEST_WIDTH_TO_STRIDE(w, px_size)`

### Functions

`void lv_test_wait(uint32_t ms)`

Emulate a delay. It's not real delay, but it tricks LVGL to think that the required time has been elapsed. `lv_timer_handler` is called after each millisecond, meaning all the events will be fired inside this function. At the end the animations and display will be also updated.

#### Parametri

`ms` -- the number of milliseconds to pass

`void lv_test_fast_forward(uint32_t ms)`

Emulates some time passing. Update the animations and the display only once at the end.

#### Parametri

`ms` -- the number of milliseconds to pass

`static inline size_t lv_test_get_free_mem(void)`

**lv\_test\_indev.h**

### Functions

`void lv_test_indev_create_all(void)`

Create a mouse (pointer), keypad, and encoder indevs. They can be controlled via function calls during the test

`void lv_test_indev_delete_all(void)`

Delete all test input devices

`lv_indev_t *lv_test_indev_get_indev(lv_indev_type_t type)`

Get one of the indev created in `lv_test_indev_create_all`

#### Parametri

`type` -- type of the indev to get

#### Ritorna

the indev

`void lv_test_mouse_move_to(int32_t x, int32_t y)`

Move the mouse to the given coordinates. This function doesn't wait, but just changes the state and returns immediately.

#### Parametri

- `x` -- the target absolute X coordinate
- `y` -- the target absolute Y coordinate

`void lv_test_mouse_move_to_obj(lv_obj_t *obj)`

Move the mouse to the center of a widget This function doesn't wait, but just changes the state and returns immediately.

**Parametri**

**obj** -- pointer to an widget

**void lv\_test\_mouse\_move\_by(int32\_t x, int32\_t y)**

Move the mouse cursor. Keep the pressed or released state This function doesn't wait, but just changes the state and returns immediately.

**Parametri**

- **x** -- the difference in X to move
- **y** -- the difference in Y to move

**void lv\_test\_mouse\_press(void)**

Make the mouse button pressed. This function doesn't wait, but just changes the state and returns immediately.

**void lv\_test\_mouse\_release(void)**

Make the mouse button released. This function doesn't wait, but just changes the state and returns immediately.

**void lv\_test\_mouse\_click\_at(int32\_t x, int32\_t y)**

Emulate a click on a given point. First set the released state, wait a little, press, wait, and release again. The wait time is 50ms. Internally `lv_timer_handler` is called, meaning all the events will be fired inside this function.

**Parametri**

- **x** -- the target absolute X coordinate
- **y** -- the target absolute Y coordinate

**void lv\_test\_key\_press(uint32\_t k)**

Emulate a key press. This function doesn't wait, but just changes the state and returns immediately.

**Parametri**

**k** -- the key to press

**void lv\_test\_key\_release(void)**

Release the previously press key. This function doesn't wait, but just changes the state and returns immediately.

**Parametri**

**k** -- the key to press

**void lv\_test\_key\_hit(uint32\_t k)**

Emulate a key hit. First set the released state, wait a little, press, wait, and release again. The wait time is 50ms. Internally `lv_timer_handler` is called, meaning all the events will be fired inside this function.

**Parametri**

**k** -- the key to hit

**void lv\_test\_encoder\_add\_diff(int32\_t d)**

Emulate encoder rotation, use positive parameter to rotate to the right and negative to rotate to the left. This function doesn't wait, but just changes the state and returns immediately.

**Parametri**

**d** -- number of encoder ticks to emulate

**void lv\_test\_encoder\_turn(int32\_t d)**

Emulate an encoder turn a wait 50ms. Use positive parameter to rotate to the right and negative to rotate to the left. Internally `lv_timer_handler` is called, meaning all the events will be fired inside this function.

**Parametri**

**d** -- number of encoder ticks to emulate

**void lv\_test\_encoder\_press(void)**

Emulate an encoder press. This function doesn't wait, but just changes the state and returns immediately.

**void lv\_test\_encoder\_release(void)**

Emulate an encoder release. This function doesn't wait, but just changes the state and returns immediately.

**void lv\_test\_encoder\_click(void)**

Emulate am encoder click. First set the released state, wait a little, press, wait, and release again. The wait time is 50ms. Internally `lv_timer_handler` is called, meaning all the events will be fired inside this function.

**lv\_test\_indev\_gesture.h****Functions**

**void lv\_test\_indev\_gesture\_create(void)**

Create a touch (pointer) indevs. They can be controlled via function calls during the test

**lv\_indev\_t \*lv\_test\_indev\_get\_gesture\_indev(lv\_indev\_type\_t type)**

Get one of the indev created in `lv_test_indev_gesture_create`

**Parametri**

- **type** -- type of the indev to get

**Ritorna**

the indev

**void lv\_test\_gesture\_set\_pinch\_data(lv\_point\_t point\_0, lv\_point\_t point\_1)**

Set two touch points data for pinch gesture

**Parametri**

- **point\_0** -- First touch point coordinates
- **point\_1** -- Second touch point coordinates

**void lv\_test\_gesture\_pinch\_press(void)**

Trigger press state of pinch gesture (both touch points pressed)

**void lv\_test\_gesture\_pinch\_release(void)**

Trigger release state of pinch gesture (both touch points released)

**void lv\_test\_gesture\_pinch(lv\_point\_t point\_begin\_0, lv\_point\_t point\_begin\_1, lv\_point\_t point\_end\_0, lv\_point\_t point\_end\_1)**

Simulate a complete pinch gesture operation

**Parametri**

- **point\_begin\_0** -- Starting coordinates of first touch point
- **point\_begin\_1** -- Starting coordinates of second touch point
- **point\_end\_0** -- Ending coordinates of first touch point
- **point\_end\_1** -- Ending coordinates of second touch point

**lv\_test\_private.h**

struct **lv\_test\_state\_t**

**Public Members**

lv\_indev\_t \***mouse\_indev**

lv\_indev\_t \***keypad\_indev**

lv\_indev\_t \***encoder\_indev**

lv\_draw\_buf\_t **draw\_buf**

int32\_t **x\_act**

int32\_t **y\_act**

uint32\_t **key\_act**

```

int32_t diff_act

bool mouse_pressed

bool key_pressed

bool enc_pressed

lv_indev_t *gesture_indev

lv_indev_touch_data_t *touch_data

uint8_t max_touch_cnt

```

## lv\_test\_screenshot\_compare.h

### Functions

bool **lv\_test\_screenshot\_compare**(const char \*fn\_ref)

Compare the current content of the test screen with a reference PNG image

- If the reference image is not found it will be created automatically from the rendered screen.
- If the compare fails an <image\_name>\_err.png file will be created with the rendered content next to the reference image.

It requires libPNG.

#### Nota

This function assumes that the default display is the test display that was created by `lv_test_display_create()`

#### Parametri

**fn\_ref** -- path to the reference image. Will be appended toREF\_IMGS\_PATH if set.

#### Ritorna

true: the reference image and the display are the same; false: they are different (<image\_name>\_err.png is created).

## 14.20.12 translation

### lv\_translation.h

### Functions

void **lv\_translation\_init**(void)

Initialize the translation module

void **lv\_translation\_deinit**(void)

De-initialize the translation module and free all allocated translations

lv\_translation\_pack\_t \***lv\_translation\_add\_static**(const char \*languages[], const char \*tags[], const char \*translations[])

Register a translation pack from static arrays. All the pointers need to be static, that is to live while they are used

#### Parametri

- **languages** -- List of languages. E.g. {"en", "de", NULL}
- **tags** -- Tags that are using in the UI. E.g. {"dog", "cat", NULL}

- **translations** -- List of translations. E.g. {"Dog", "Cat", "Hund", "Katze"}

**Ritorna**

The created pack

`lv_translation_pack_t *lv_translation_add_dynamic(void)`

Add a pack to which translations can be added dynamically. `pack->languages` needs to be a malloc-ed array where each language is also malloc-ed as an element. `pack->translation_array` stores the translation having `lv_translation_tag_dsc_t` items In each array element `tag` is a malloced string, `translations` is a malloc-ed array with malloc-ed array for each element.

**Ritorna**

the created pack to which data can be added manually.

`void lv_translation_set_language(const char *lang)`

Select the current language

**Parametri**

`lang` -- a string from the defined languages. E.g. "en" or "de"

`const char *lv_translation_get(const char *tag)`

Get the translated version of a tag on the selected language

 **Nota**

fallback rules:

- if the tag is found on the selected language return it
- if the tag is not found on the selected language, use the fist language
- if the tag is not found on the first language, return the tag

**Parametri**

`tag` -- the tag to translate

**Ritorna**

the translation

`static inline const char *lv_tr(const char *tag)`

Shorthand of `lv_translation_set_language`

**Parametri**

`tag` -- the tag to translate

**Ritorna**

the translation

`lv_result_t lv_translation_add_language(lv_translation_pack_t *pack, const char *lang)`

Add a new language to a dynamic language pack. All languages should be added before adding tags

**Parametri**

- `pack` -- pointer to a dynamic translation pack
- `lang` -- language to add, e.g. "en", or "de"

**Ritorna**

`LV_RESULT_OK`: success, `LV_RESULT_INVALID`: failed

`int32_t lv_translation_get_language_index(lv_translation_pack_t *pack, const char *lang_name)`

Get the index of a language in a pack.

**Parametri**

- `pack` -- pointer to a static or dynamic language pack
- `lang_name` -- name of the language to find

**Ritorna**

index of the language or -1 if not found.

`lv_translation_tag_dsc_t *lv_translation_add_tag(lv_translation_pack_t *pack, const char *tag_name)`

Add a new tag to a dynamic language pack. Once the tag is added the translations for each language can be added too by using `lv_translation_set_tag_translation`

#### Parametri

- **pack** -- pointer to a dynamic translation pack
- **tag\_name** -- name of the tag, e.g. "dog", or "house"

#### Ritorna

pointer to the allocated tag descriptor

`lv_result_t lv_translation_set_tag_translation(lv_translation_pack_t *pack,  
 lv_translation_tag_dsc_t *tag, uint32_t lang_idx,  
 const char *trans)`

Add a translation to a tag in a dynamic translation pack

#### Parametri

- **pack** -- pointer to a dynamic translation pack
- **tag** -- return value of `lv_translation_add_tag`
- **lang\_idx** -- index of the language for which translation should be set
- **trans** -- the translation on the given language

#### Ritorna

LV\_RESULT\_OK: success, LV\_RESULT\_INVALID: failed

## lv\_translation\_private.h

`struct _lv_translation_tag_dsc_t`

#### Public Members

`const char *tag`

`const char **translations`

Translations for each language

`struct _lv_translation_pack_t`

#### Public Members

`const char **languages`

`uint32_t language_cnt`

`uint32_t is_static`

`const char **tag_p`

`const char **translation_p`

`lv_array_t translation_array`

## 14.20.13 xml

[lv\\_xml.h](#)

### Defines

**LV\_XML\_MAX\_PATH\_LENGTH**

### Functions

`void lv_xml_init(void)`

`void lv_xml_deinit(void)`

`void *lv_xml_create(lv_obj_t *parent, const char *name, const char **attrs)`

`void *lv_xml_create_in_scope(lv_obj_t *parent, lv_xml_component_scope_t *parent_ctx,  
 lv_xml_component_scope_t *scope, const char **attrs)`

`void lv_xml_set_default_asset_path(const char *path_prefix)`

Set a path to prefix the image and font file source paths.

In globals.xml usually the source path is like "images/logo.png". But on the actual device it can be located at e.g. "A:ui/assets/images/logo.png". By setting "A:ui/assets/" the path set in the XML files will be prefixed accordingly.

#### Parametri

**path\_prefix** -- the path to be used as prefix

`lv_result_t lv_xml_register_font(lv_xml_component_scope_t *scope, const char *name, const lv_font_t  
 *font)`

`const lv_font_t *lv_xml_get_font(lv_xml_component_scope_t *scope, const char *name)`

`lv_result_t lv_xml_register_image(lv_xml_component_scope_t *scope, const char *name, const void  
 *src)`

`const void *lv_xml_get_image(lv_xml_component_scope_t *scope, const char *name)`

`lv_result_t lv_xml_register_subject(lv_xml_component_scope_t *scope, const char *name, lv_subject_t  
 *subject)`

Map globally available subject name to an actual subject variable

#### Parametri

- **name** -- name of the subject
- **subject** -- pointer to a subject

#### Ritorna

`LV_RESULT_OK`: success

`lv_subject_t *lv_xml_get_subject(lv_xml_component_scope_t *scope, const char *name)`

Get a subject by name.

#### Parametri

- **scope** -- If specified start searching in that component's subject list, and if not found search in the global space. If NULL search in global space immediately.
- **name** -- Name of the subject to find.

#### Ritorna

Pointer to the subject or NULL if not found.

`lv_result_t lv_xml_register_const(lv_xml_component_scope_t *scope, const char *name, const char  
 *value)`

`const char *lv_xml_get_const(lv_xml_component_scope_t *scope, const char *name)`

```
lv_result_t lv_xml_register_event_cb(lv_xml_component_scope_t *scope, const char *name,
                                         lv_event_cb_t cb)
```

```
lv_event_cb_t lv_xml_get_event_cb(lv_xml_component_scope_t *scope, const char *name)
```

```
lv_result_t lv_xml_register_timeline(lv_xml_component_scope_t *scope, const char *name)
```

```
void *lv_xml_get_timeline(lv_xml_component_scope_t *scope, const char *name)
```

## lv\_xml\_base\_types.h

### Functions

*lv\_state\_t* **lv\_xml\_state\_to\_enum**(const char \*txt)

Convert an state string to enum

**Parametri**

**txt** -- e.g. "pressed"

**Ritorna**

the related enum, e.g. LV\_STATE\_PRESSED

int32\_t **lv\_xml\_to\_size**(const char \*txt)

Process inputs "content", "32", "32px", or "25%" and convert them to integer

**Parametri**

**txt** -- the input string

**Ritorna**

the integer size

*lv\_align\_t* **lv\_xml\_align\_to\_enum**(const char \*txt)

Convert an align string to enum

**Parametri**

**txt** -- e.g. "center"

**Ritorna**

the related enum, e.g. LV\_ALIGN\_CENTER

*lv\_dir\_t* **lv\_xml\_dir\_to\_enum**(const char \*txt)

Convert a direction string to enum

**Parametri**

**txt** -- e.g. "top"

**Ritorna**

the related enum, e.g. LV\_DIR\_TOP

*lv\_border\_side\_t* **lv\_xml\_border\_side\_to\_enum**(const char \*txt)

Convert a direction string to enum

**Parametri**

**txt** -- e.g. "top"

**Ritorna**

the related enum, e.g. LV\_BORDER\_SIDE\_TOP

*lv\_base\_dir\_t* **lv\_xml\_base\_dir\_to\_enum**(const char \*txt)

Convert a base dir string to enum

**Parametri**

**txt** -- e.g. "rtl"

**Ritorna**

the related enum, e.g. LV\_BASE\_DIR\_RTL

*lv\_grad\_dir\_t* **lv\_xml\_grad\_dir\_to\_enum**(const char \*txt)

Convert a grad dir string to enum

**Parametri**

**txt** -- e.g. "hor"

**Ritorna**

the related enum, e.g. LV\_GRAD\_DIR\_HOR

*lv\_text\_align\_t* **lv\_xml\_text\_align\_to\_enum**(const char \*txt)

Convert a text align string to enum

**Parametri**

**txt** -- e.g. "left"

**Ritorna**

the related enum, e.g. LV\_TEXT\_ALIGN\_LEFT

*lv\_text\_decor\_t* **lv\_xml\_text\_decor\_to\_enum**(const char \*txt)

Convert a text decor string to enum

**Parametri**

**txt** -- e.g. "underline"

**Ritorna**

the related enum, e.g. LV\_TEXT\_DECOR\_UNDERLINE

*lv\_scroll\_snap\_t* **lv\_xml\_scroll\_snap\_to\_enum**(const char \*txt)

Convert a scroll snap string to enum

**Parametri**

**txt** -- e.g. "start"

**Ritorna**

the related enum, e.g. LV\_SCROLL\_SNAP\_START

*lv\_flex\_flow\_t* **lv\_xml\_flex\_flow\_to\_enum**(const char \*txt)

Convert a flex flow string to enum

**Parametri**

**txt** -- e.g. "row\_wrap"

**Ritorna**

the related enum, e.g. LV\_FLEX\_FLOW\_ROW\_WRAP

*lv\_flex\_align\_t* **lv\_xml\_flex\_align\_to\_enum**(const char \*txt)

Convert a flex align string to enum

**Parametri**

**txt** -- e.g. "space\_between"

**Ritorna**

the related enum, e.g. LV\_FLEX\_ALIGN\_SPACE\_BETWEEN

*lv\_grid\_align\_t* **lv\_xml\_grid\_align\_to\_enum**(const char \*txt)

Convert a grid align string to enum

**Parametri**

**txt** -- e.g. "space\_between"

**Ritorna**

the related enum, e.g. LV\_GRID\_ALIGN\_SPACE\_BETWEEN

*lv\_layout\_t* **lv\_xml\_layout\_to\_enum**(const char \*txt)

Convert a layout string to enum

**Parametri**

**txt** -- e.g. "flex"

**Ritorna**

the related enum, e.g. LV\_LAYOUT\_FLEX

*lv\_blend\_mode\_t* **lv\_xml\_blend\_mode\_to\_enum**(const char \*txt)

Convert a blend mode string to enum

**Parametri**

**txt** -- e.g. "additive"

**Ritorna**

the related enum, e.g. LV\_BLEND\_MODE\_ADDITIVE

*lv\_event\_code\_t* **lv\_xml\_trigger\_text\_to\_enum\_value**(const char \*txt)

Convert an event trigger string to enum

**Parametri**

**txt** -- e.g. "clicked"

**Ritorna**

the related enum, e.g. LV\_EVENT\_CLICKED

`lv_screen_load_anim_t lv_xml_screen_load_anim_text_to_enum_value(const char *txt)`

Convert a screen animation type string to enum

**Parametri**

**txt** -- e.g. "over\_right"

**Ritorna**

the related enum, e.g. LV\_SCREEN\_LOAD\_ANIM\_OVER\_RIGHT

`lv_style_prop_t lv_xml_style_prop_to_enum(const char *txt)`

Convert a style property string to enum

**Parametri**

**txt** -- e.g. "bg\_color"

**Ritorna**

the related enum, e.g. LV\_STYLE\_BG\_COLOR or LV\_STYLE\_PROP\_INV if not found.

`lv_state_t lv_xml_style_state_to_enum(const char *txt)`

Convert a style state to enum

**Parametri**

**txt** -- e.g. "pressed"

**Ritorna**

the enum LV\_STATE\_PRESSED

`lv_part_t lv_xml_style_part_to_enum(const char *txt)`

Convert a style part to enum

**Parametri**

**txt** -- e.g. "knob"

**Ritorna**

the enum LV\_PART\_KNOB

`lv_style_selector_t lv_xml_style_selector_text_to_enum(const char *str)`

Convert ORed style parts and states to an ORed selector

**Parametri**

**txt** -- e.g. "knob|pressed"

**Ritorna**

the enum LV\_PART\_KNOB | LV\_STATE\_PRESSED

**lv\_xml\_component.h****Functions**

`lv_obj_t *lv_xml_component_process(lv_xml_parser_state_t *state, const char *name, const char **attrs)`

Process a component during parsing an XML. It create a widget and apply all the attributes

**Parametri**

- **state** -- the current parsing state
- **name** -- name of the component
- **attrs** -- attributes of the widget

**Ritorna**

`lv_result_t lv_xml_component_register_from_data(const char *name, const char *xml_def)`

Load the styles, constants, another data of the component. It needs to be called only once for each component.

**Parametri**

- **name** -- the name as the component will be referenced later in other components
- **xml\_def** -- the XML definition of the component as a NULL terminated string

**Ritorna**

LV\_RES\_OK: loaded successfully, LV\_RES\_INVALID: otherwise

`lv_result_t lv_xml_component_register_from_file(const char *path)`

Load the styles, constants, another data of the component. It needs to be called only once for each component.

**Parametri**

**path** -- path to an XML file

**Ritorna**

LV\_RES\_OK: loaded successfully, LV\_RES\_INVALID: otherwise

`lv_xml_component_scope_t *lv_xml_component_get_scope(const char *component_name)`

Get the scope of a component which was registered by `lv_xml_component_register_from_data` or `lv_xml_component_register_from_file`

**Parametri**

**component\_name** -- name of the component

**Ritorna**

pointer the scope or NULL if not found

`lv_result_t lv_xml_component_unregister(const char *name)`

Remove a component from from the list.

**Parametri**

**name** -- the name of the component (used during registration)

**Ritorna**

LV\_RESULT\_OK on successful unregistration, LV\_RESULT\_INVALID otherwise.

## lv\_xml\_component\_private.h

### Typedef

```
typedef void *(*lv_xml_component_process_cb_t)(lv_obj_t *parent, const char *data, const char **attrs)
```

### Functions

`void lv_xml_component_init(void)`

Initialize the components system.

`void lv_xml_component_scope_init(lv_xml_component_scope_t *scope)`

Initialize the linked lists of a component context

**Parametri**

**scope** -- pointer to a component contexts

`struct _lv_xml_component_scope_t`

### Public Members

`const char *name`

`lv_ll_t style_ll`

`lv_ll_t const_ll`

`lv_ll_t param_ll`

`lv_ll_t gradient_ll`

`lv_ll_t subjects_ll`

*lv\_ll\_t* **timeline\_ll**

*lv\_ll\_t* **font\_ll**

*lv\_ll\_t* **image\_ll**

*lv\_ll\_t* **event\_ll**

const char \***view\_def**

const char \***extends**

uint32\_t **is\_widget**

uint32\_t **is\_screen**

struct *lv\_xml\_component\_scope\_t* \***next**

struct **lv\_xml\_const\_t**

#### Public Members

const char \***name**

const char \***value**

struct **lv\_xml\_subject\_t**

#### Public Members

const char \***name**

*lv\_subject\_t* \***subject**

struct **lv\_xml\_timeline\_t**

#### Public Members

const char \***name**

*lv\_ll\_t* **anim\_ll**

struct **lv\_xml\_param\_t**

**Public Members**`const char *name``const char *def``const char *type``struct lv_xml_grad_t`**Public Members**`const char *name``lv_grad_dsc_t grad_dsc``lv_xml_parser.h`**Enums**`enum lv_xml_parser_section_t`

*Values:*

`enumerator LV_XML_PARSER_SECTION_NONE``enumerator LV_XML_PARSER_SECTION_API``enumerator LV_XML_PARSER_SECTION_CONSTS``enumerator LV_XML_PARSER_SECTION_GRAD``enumerator LV_XML_PARSER_SECTION_GRAD_STOP``enumerator LV_XML_PARSER_SECTION_STYLES``enumerator LV_XML_PARSER_SECTION_FONTS``enumerator LV_XML_PARSER_SECTION_IMAGES``enumerator LV_XML_PARSER_SECTION_SUBJECTS``enumerator LV_XML_PARSER_SECTION_ANIMATION``enumerator LV_XML_PARSER_SECTION_TIMELINE``enumerator LV_XML_PARSER_SECTION_VIEW`

**Functions**

```
void lv_xml_parser_state_init(lv_xml_parser_state_t *state)
void lv_xml_parser_start_section(lv_xml_parser_state_t *state, const char *name)
void lv_xml_parser_end_section(lv_xml_parser_state_t *state, const char *name)
void *lv_xml_state_get_parent(lv_xml_parser_state_t *state)
void *lv_xml_state_get_item(lv_xml_parser_state_t *state)

struct _lv_xml_parser_state_t
```

**Public Members**

```
const char *tag_name
lv_xml_component_scope_t scope
lv_ll_t parent_ll
lv_obj_t *parent
lv_obj_t *item
lv_obj_t *view
void *context
const char **parent_attrs
lv_xml_component_scope_t *parent_scope
lv_xml_parser_section_t section
```

**lv\_xml\_private.h**

```
struct lv_xml_font_t
```

**Public Members**

```
const char *name
const lv_font_t *font
void (*font_destroy_cb)(lv_font_t*)
```

```
struct lv_xml_image_t
```

**Public Members**

```
const char *name
```

```
const void *src
```

```
struct lv_xml_event_cb_t
```

**Public Members**

```
const char *name
```

```
lv_event_cb_t cb
```

**lv\_xml\_style.h**

**Typedef**

```
typedef struct _lv_xml_style_t lv_xml_style_t
```

**Functions**

lv\_result\_t **lv\_xml\_style\_register**(lv\_xml\_component\_scope\_t \*scope, const char \*\*attrs)

Add a style to **ctx** and set the style properties from **attrs**

**Parametri**

- **scope** -- add styles here. (Constants should be already added as style properties might use them)
- **attrs** -- list of attribute names and values

const char \***lv\_xml\_style\_string\_process**(char \*txt, *lv\_style\_selector\_t* \*selector)

Decompose a string like "style1:pressed:checked:knob" to style name and selector

**Parametri**

- **txt** -- the input string
- **selector** -- store the selectors here

**Ritorna**

the style name or **NULL** on any error

*lv\_xml\_style\_t* \***lv\_xml\_get\_style\_by\_name**(lv\_xml\_component\_scope\_t \*scope, const char \*name)

Find a style by name which was added by **lv\_xml\_style\_register**

**Parametri**

- **scope** -- the default context to search in
- **name** -- the name of the style. Can start with a component name prefix (e.g. `my_button.blue`) to overwrite the ctx

**Ritorna**

the style structure

*lv\_grad\_dsc\_t* \***lv\_xml\_component\_get\_grad**(lv\_xml\_component\_scope\_t \*scope, const char \*name)

Get a gradient descriptor defined for a component

**Parametri**

- **scope** -- component context where the gradient should be found
- **name** -- name of the gradient

**Ritorna**

a gradient descriptor

```
struct _lv_xml_style_t
```

**Public Members**

const char \***name**

const char \***long\_name**

*lv\_style\_t* **style**

**lv\_xml\_test.h**

**Enums**

enum **lv\_xml\_test\_step\_type\_t**

*Values:*

enumerator **LV\_XML\_TEST\_STEP\_TYPE\_NONE**

enumerator **LV\_XML\_TEST\_STEP\_TYPE\_MOVE\_TO**

enumerator **LV\_XML\_TEST\_STEP\_TYPE\_PRESS**

enumerator **LV\_XML\_TEST\_STEP\_TYPE\_RELEASE**

enumerator **LV\_XML\_TEST\_STEP\_TYPE\_CLICK\_AT**

enumerator **LV\_XML\_TEST\_STEP\_TYPE\_WAIT**

enumerator **LV\_XML\_TEST\_STEP\_TYPE\_FREEZE**

enumerator **LV\_XML\_TEST\_STEP\_TYPE\_SCREENSHOT\_COMPARE**

enumerator **LV\_XML\_TEST\_STEP\_TYPE SUBJECT\_SET**

enumerator **LV\_XML\_TEST\_STEP\_TYPE SUBJECT\_COMPARE**

**Functions**

lv\_result\_t **lv\_xml\_test\_register\_from\_data**(const char \*xml\_def, const char  
\*ref\_image\_path\_prefix)

Load the styles, constants, another data of the test. It needs to be called only once for each test.

**Parametri**

- **xml\_def** -- the XML definition of the test as a NULL terminated string
- **ref\_image\_path\_prefix** -- prefix for the path of reference images

**Ritorna**

LV\_RES\_OK: loaded successfully, LV\_RES\_INVALID: otherwise

lv\_result\_t **lv\_xml\_test\_register\_from\_file**(const char \*path, const char \*ref\_image\_path\_prefix)

Load the styles, constants, another data of the test. It needs to be called only once for each test.

**Parametri**

- **path** -- path to an XML file
- **ref\_image\_path\_prefix** -- prefix for the path of reference images

**Ritorna**  
LV\_RES\_OK: loaded successfully, LV\_RES\_INVALID: otherwise

**void lv\_xml\_test\_unregister(void)**  
Free resources allocated for testing.

**void lv\_xml\_test\_run\_init(void)**  
Switch to testing mode. Needs to be called to use `lv_xml_test_run_next()`

**bool lv\_xml\_test\_run\_next(uint32\_t slowdown)**  
Run the next test step.  
**Parametri**  
**slowdown** -- 0: max speed, 1: real speed, 2: half speed, ... ,10: ten times slower  
**Ritorna**  
true: the step passed; false: the step failed

**void lv\_xml\_test\_run\_stop(void)**  
Leave testing mode.

**uint32\_t lv\_xml\_test\_run\_all(uint32\_t slowdown)**  
Run all the test steps. It calls `lv_xml_test_run_init()`, `lv_xml_test_run_next()`, and `lv_xml_test_stop` internally so no further preparation or cleanup is needed.  
**Parametri**  
**slowdown** -- 0: max speed, 1: real speed, 2: half speed, ... ,10: ten times slower  
**Ritorna**  
number of failed tests

**uint32\_t lv\_xml\_test\_get\_step\_count(void)**  
Get the number of steps in a test  
**Ritorna**  
the number of <step>s

**lv\_xml\_test\_step\_type\_t lv\_xml\_test\_get\_step\_type(uint32\_t idx)**  
Get the type of a step  
**Parametri**  
**idx** -- the index of a step (< step\_count)  
**Ritorna**  
element of `lv_xml_test_step_type_t`

**bool lv\_xml\_test\_get\_status(uint32\_t idx)**  
Check if the a step was passed. Can be called after `lv_xml_test_run()`  
**Parametri**  
**idx** -- the index of a step (< step\_count)  
**Ritorna**  
true: the step passed, false: the step failed

## lv\_xml\_translation.h

### Functions

**lv\_result\_t lv\_xml\_translation\_register\_from\_file(const char \*path)**

Register translations from an XML file

**Parametri**  
**path** -- path to an XML file (staring with a driver letter)

**Ritorna**  
LV\_RES\_OK: no error

**lv\_result\_t lv\_xml\_translation\_register\_from\_data(const char \*xml\_def)**

Register translations from an XML string

**Parametri**  
**xml\_def** -- the XML definition as a string

**Ritorna**

LV\_RES\_OK: no error

**lv\_xml\_update.h****Functions**lv\_result\_t **lv\_xml\_update\_from\_data**(const char \*xml\_def)

Load the styles, constants, another data of the component. It needs to be called only once for each component.

**Parametri**• **xml\_def** -- the XML definition of the component as a NULL terminated string**Ritorna**

LV\_RES\_OK: loaded successfully, LV\_RES\_INVALID: otherwise

**lv\_xml\_utils.h****Functions**const char \***lv\_xml\_get\_value\_of**(const char \*\*attrs, const char \*name)int32\_t **lv\_xml\_atoi**(const char \*str)int32\_t **lv\_xml\_atoi\_split**(const char \*\*str, char delimiter)Convert sections of a string to int. The end of the string is indicated by the **delimiter**.**Parametri**

- **str** -- pointer to a string, it will point to the character after the delimiter
- **delimiter** -- a character to indicate the end of the int

**Ritorna**

the int before the next delimiter

float **lv\_xml\_atof**(const char \*str)float **lv\_xml\_atof\_split**(const char \*\*str, char delimiter)Convert sections of a string to float. The end of the string is indicated by the **delimiter**.**Parametri**

- **str** -- pointer to a string, it will point to the character after the delimiter
- **delimiter** -- a character to indicate the end of the float

**Ritorna**

the float before the next delimiter

lv\_color\_t **lv\_xml\_to\_color**(const char \*str)lv\_opa\_t **lv\_xml\_to\_opa**(const char \*str)

Concert percentage or integer opacity value from string to integer.

**Parametri**• **str** -- e.g. "70%" or 180**Ritorna**

0..255

bool **lv\_xml\_to\_bool**(const char \*str)int32\_t **lv\_xml\_strtol**(const char \*str, char \*\*endptr, int32\_t base)char \***lv\_xml\_split\_str**(char \*\*src, char delimiter)

Find a delimiter in a string, terminate the string on the delimiter and update the source string to the next part

**Parametri**

- **src** -- point to a variable containing the input string
- **delimiter** -- a delimiter character, e.g. ':'

**Ritorna**

the beginning of next section in the string closed at the delimiter

**lv\_xml\_widget.h****Typedef**

```
typedef void *(*lv_xml_widget_create_cb_t)(lv_xml_parser_state_t *state, const char **parentAttrs)
```

```
typedef void (*lv_xml_widget_apply_cb_t)(lv_xml_parser_state_t *state, const char **parentAttrs)
```

```
typedef struct _lv_widget_processor_t lv_widget_processor_t
```

**Functions**

```
lv_result_t lv_xml_widget_register(const char *name, lv_xml_widget_create_cb_t createCb,  
                                lv_xml_widget_apply_cb_t applyCb)
```

```
lv_widget_processor_t *lv_xml_widget_get_processor(const char *name)
```

```
lv_widget_processor_t *lv_xml_widget_get_extended_widget_processor(const char *extends)
```

```
struct lv_widget_processor_t
```

**Public Members**

```
const char *name
```

```
lv_xml_widget_create_cb_t create_cb
```

```
lv_xml_widget_apply_cb_t apply_cb
```

```
struct _lv_widget_processor_t *next
```

**parsers****lv\_xml\_arc\_parser.h****Functions**

```
void *lv_xml_arc_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_arc_apply(lv_xml_parser_state_t *state, const char **attrs)
```

**lv\_xml\_bar\_parser.h****Functions**

```
void *lv_xml_bar_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_bar_apply(lv_xml_parser_state_t *state, const char **attrs)
```

**lv\_xml\_button\_parser.h****Functions**

```
void *lv_xml_button_create(lv_xml_parser_state_t *state, const char **attrs)
```

```
void lv_xml_button_apply(lv_xml_parser_state_t *state, const char **attrs)
```

## lv\_xml\_buttonmatrix\_parser.h

### Functions

```
void *lv_xml_buttonmatrix_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_buttonmatrix_apply(lv_xml_parser_state_t *state, const char **attrs)
```

## lv\_xml\_calendar\_parser.h

### Functions

```
void *lv_xml_calendar_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_calendar_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_calendar_header_arrow_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_calendar_header_arrow_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_calendar_header_dropdown_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_calendar_header_dropdown_apply(lv_xml_parser_state_t *state, const char **attrs)
```

## lv\_xml\_canvas\_parser.h

### Functions

```
void *lv_xml_canvas_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_canvas_apply(lv_xml_parser_state_t *state, const char **attrs)
```

## lv\_xml\_chart\_parser.h

### Functions

```
void *lv_xml_chart_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_chart_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_chart_series_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_chart_series_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_chart_cursor_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_chart_cursor_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_chart_axis_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_chart_axis_apply(lv_xml_parser_state_t *state, const char **attrs)
```

## lv\_xml\_checkbox\_parser.h

### Functions

```
void *lv_xml_checkbox_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_checkbox_apply(lv_xml_parser_state_t *state, const char **attrs)
```

**lv\_xml\_dropdown\_parser.h****Functions**

```
void *lv_xml_dropdown_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_dropdown_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_dropdown_list_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_dropdown_list_apply(lv_xml_parser_state_t *state, const char **attrs)
```

**lv\_xml\_image\_parser.h****Functions**

```
void *lv_xml_image_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_image_apply(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_check_file(const char *filepath)
```

**lv\_xml\_keyboard\_parser.h****Functions**

```
void *lv_xml_keyboard_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_keyboard_apply(lv_xml_parser_state_t *state, const char **attrs)
```

**lv\_xml\_label\_parser.h****Functions**

```
void *lv_xml_label_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_label_apply(lv_xml_parser_state_t *state, const char **attrs)
```

**lv\_xml\_obj\_parser.h****Functions**

```
void *lv_xml_obj_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_obj_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_style_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_style_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_remove_style_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_remove_style_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_remove_style_all_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_remove_style_all_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_event_cb_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_event_cb_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_subject_set_create(lv_xml_parser_state_t *state, const char **attrs)
```

```

void lv_obj_xml_subject_set_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_subject_increment_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_subject_increment_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_bind_style_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_bind_style_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_bind_flag_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_bind_flag_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_bind_state_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_bind_state_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_screen_load_event_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_screen_load_event_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_screen_create_event_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_screen_create_event_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_obj_xml_play_timeline_event_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_obj_xml_play_timeline_event_apply(lv_xml_parser_state_t *state, const char **attrs)

```

## lv\_xml\_roller\_parser.h

### Functions

```

void *lv_xml_roller_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_roller_apply(lv_xml_parser_state_t *state, const char **attrs)

```

## lv\_xml\_scale\_parser.h

### Functions

```

void *lv_xml_scale_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_scale_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_scale_section_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_scale_section_apply(lv_xml_parser_state_t *state, const char **attrs)

```

## lv\_xml\_slider\_parser.h

### Functions

```

void *lv_xml_slider_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_slider_apply(lv_xml_parser_state_t *state, const char **attrs)

```

**lv\_xml\_spangroup\_parser.h****Functions**

```
void *lv_xml_spangroup_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_spangroup_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_spangroup_span_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_spangroup_span_apply(lv_xml_parser_state_t *state, const char **attrs)
```

**lv\_xml\_table\_parser.h****Functions**

```
void *lv_xml_table_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_table_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_table_column_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_table_column_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_table_cell_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_table_cell_apply(lv_xml_parser_state_t *state, const char **attrs)
```

**lv\_xml\_tabview\_parser.h****Functions**

```
void *lv_xml_tabview_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_tabview_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_tabview_tab_bar_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_tabview_tab_bar_apply(lv_xml_parser_state_t *state, const char **attrs)
void *lv_xml_tabview_tab_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_tabview_tab_apply(lv_xml_parser_state_t *state, const char **attrs)
```

**lv\_xml\_textarea\_parser.h****Functions**

```
void *lv_xml_textarea_create(lv_xml_parser_state_t *state, const char **attrs)
void lv_xml_textarea_apply(lv_xml_parser_state_t *state, const char **attrs)
```

**14.21 stdlib****14.21.1 lv\_mem.h****Typedef**

typedef void \*lv\_mem\_pool\_t

## Functions

`void lv_mem_init(void)`

Initialize to use malloc/free/realloc etc

`void lv_mem_deinit(void)`

Drop all dynamically allocated memory and reset the memory pools' state

`lv_mem_pool_t lv_mem_add_pool(void *mem, size_t bytes)`

`void lv_mem_remove_pool(lv_mem_pool_t pool)`

`void *lv_malloc(size_t size)`

Allocate memory dynamically

**Parametri**

**size** -- requested size in bytes

**Ritorna**

pointer to allocated uninitialized memory, or NULL on failure

`void *lv_calloc(size_t num, size_t size)`

Allocate a block of zeroed memory dynamically

**Parametri**

- **num** -- requested number of element to be allocated.
- **size** -- requested size of each element in bytes.

**Ritorna**

pointer to allocated zeroed memory, or NULL on failure

`void *lv_zalloc(size_t size)`

Allocate zeroed memory dynamically

**Parametri**

**size** -- requested size in bytes

**Ritorna**

pointer to allocated zeroed memory, or NULL on failure

`void *lv_malloc_zeroed(size_t size)`

Allocate zeroed memory dynamically

**Parametri**

**size** -- requested size in bytes

**Ritorna**

pointer to allocated zeroed memory, or NULL on failure

`void lv_free(void *data)`

Free an allocated data

**Parametri**

**data** -- pointer to an allocated memory

`void *lv_realloc(void *data_p, size_t new_size)`

Reallocate a memory with a new size. The old content will be kept.

**Parametri**

- **data\_p** -- pointer to an allocated memory. Its content will be copied to the new memory block and freed
- **new\_size** -- the desired new size in byte

**Ritorna**

pointer to the new memory, NULL on failure

`void *lv_reallocf(void *data_p, size_t new_size)`

Reallocate a memory with a new size. The old content will be kept. In case of failure, the old pointer is free'd.

**Parametri**

- **data\_p** -- pointer to an allocated memory. Its content will be copied to the new memory block and freed
- **new\_size** -- the desired new size in byte

**Ritorna**  
pointer to the new memory, NULL on failure

**void \*lv\_malloc\_core(size\_t size)**  
Used internally to execute a plain `malloc` operation

**Parametri**

- size** -- size in bytes to `malloc`

**void lv\_free\_core(void \*p)**  
Used internally to execute a plain `free` operation

**Parametri**

- p** -- memory address to free

**void \*lv\_realloc\_core(void \*p, size\_t new\_size)**  
Used internally to execute a plain `realloc` operation

**Parametri**

- **p** -- memory address to `realloc`
- **new\_size** -- size in bytes to `realloc`

**void lv\_mem\_monitor\_core(lv\_mem\_monitor\_t \*mon\_p)**  
Used internally by `lv_mem_monitor()` to gather LVGL heap state information.

**Parametri**

- mon\_p** -- pointer to `lv_mem_monitor_t` object to be populated.

**lv\_result\_t lv\_mem\_test\_core(void)**

**lv\_result\_t lv\_mem\_test(void)**  
Tests the memory allocation system by allocating and freeing a block of memory.

**Ritorna**  
`LV_RESULT_OK` if the memory allocation system is working properly, or `LV_RESULT_INVALID` if there is an error.

**void lv\_mem\_monitor(lv\_mem\_monitor\_t \*mon\_p)**  
Give information about the work memory of dynamic allocation

**Parametri**

- mon\_p** -- pointer to a `lv_mem_monitor_t` variable, the result of the analysis will be stored here

**struct lv\_mem\_monitor\_t**  
`#include <lv_mem.h>` Heap information structure.

### Public Members

**size\_t total\_size**  
Total heap size

**size\_t free\_cnt**

**size\_t free\_size**  
Size of available memory

**size\_t free\_biggest\_size**

**size\_t used\_cnt**

**size\_t max\_used**

Max size of Heap memory used

**uint8\_t used\_pct**

Percentage used

**uint8\_t frag\_pct**

Amount of fragmentation

**14.21.2 lv\_mem\_private.h****14.21.3 lv\_sprintf.h****Defines****LV\_PRId32****LV\_PRIu32****LV\_PRIx32****LV\_PRIx32****LV\_PRIu64****LV\_PRIx64****LV\_PRIx64****Functions**

```
int lv_snprintf(char *buffer, size_t count, const char *format, ...)
```

```
int lv_vsnprintf(char *buffer, size_t count, const char *format, va_list va)
```

**14.21.4 lv\_string.h****Functions**

```
void *lv_memcpy(void *dst, const void *src, size_t len)
```

Copies a block of memory from a source address to a destination address.

**Nota**

The function does not check for any overlapping of the source and destination memory blocks.

**Parametri**

- **dst** -- Pointer to the destination array where the content is to be copied.
- **src** -- Pointer to the source of data to be copied.
- **len** -- Number of bytes to copy.

**Ritorna**

Pointer to the destination array.

**void lv\_memset( void \*dst, uint8\_t v, size\_t len )**

Fills a block of memory with a specified value.

**Parametri**

- **dst** -- Pointer to the destination array to fill with the specified value.
- **v** -- Value to be set. The value is passed as an int, but the function fills the block of memory using the unsigned char conversion of this value.
- **len** -- Number of bytes to be set to the value.

**void \*lv\_memmove( void \*dst, const void \*src, size\_t len )**

Move a block of memory from source to destination.

**Parametri**

- **dst** -- Pointer to the destination array where the content is to be copied.
- **src** -- Pointer to the source of data to be copied.
- **len** -- Number of bytes to copy

**Ritorna**

Pointer to the destination array.

**int lv\_memcmp( const void \*p1, const void \*p2, size\_t len )**

This function will compare two memory blocks.

**Parametri**

- **p1** -- Pointer to the first memory block
- **p2** -- Pointer to the second memory block
- **len** -- Number of bytes to compare

**Ritorna**

The difference between the value of the first unmatching byte.

**static inline void lv\_memzero( void \*dst, size\_t len )**

Same as `memset(dst, 0x00, len)`.

**Parametri**

- **dst** -- pointer to the destination buffer
- **len** -- number of byte to set

**size\_t lv\_strlen( const char \*str )**

Computes the length of the string str up to (but not including) the terminating null character.

**Parametri**

- **str** -- Pointer to the null-terminated byte string to be examined.

**Ritorna**

The length of the string in bytes.

**size\_t lv\_strnlen( const char \*str, size\_t max\_len )**

Computes the length of the string str up to (but not including) the terminating null character, or the given maximum length.

**Parametri**

- **str** -- Pointer to byte string that is null-terminated or at least `max_len` bytes long.
- **max\_len** -- Maximum number of characters to examine.

**Ritorna**

The length of the string in bytes.

**size\_t lv\_strlcpy( char \*dst, const char \*src, size\_t dst\_size )**

Copies up to `dst_size-1` (non-null) characters from `src` to `dst`. A null terminator is always added.

**Parametri**

- **dst** -- Pointer to the destination array where the content is to be copied.
- **src** -- Pointer to the source of data to be copied.
- **dst\_size** -- Maximum number of characters to be copied to `dst`, including the null character.

**Ritorna**

The length of `src`. The return value is equivalent to the value returned by `lv_strlen(src)`

`char *lv_strncpy(char *dst, const char *src, size_t dest_size)`

Copies up to `dest_size` characters from the string pointed to by `src` to the character array pointed to by `dst` and fills the remaining length with null bytes.

### Nota

`dst` will not be null terminated if `dest_size` bytes were copied from `src` before the end of `src` was reached.

#### Parametri

- **dst** -- Pointer to the destination array where the content is to be copied.
- **src** -- Pointer to the source of data to be copied.
- **dest\_size** -- Maximum number of characters to be copied to `dst`.

#### Ritorna

A pointer to the destination array, which is `dst`.

`char *lv_strcpy(char *dst, const char *src)`

Copies the string pointed to by `src`, including the terminating null character, to the character array pointed to by `dst`.

#### Parametri

- **dst** -- Pointer to the destination array where the content is to be copied.
- **src** -- Pointer to the source of data to be copied.

#### Ritorna

A pointer to the destination array, which is `dst`.

`int lv_strcmp(const char *s1, const char *s2)`

This function will compare two strings without specified length.

#### Parametri

- **s1** -- pointer to the first string
- **s2** -- pointer to the second string

#### Ritorna

the difference between the value of the first unmatched character.

`int lv_strncmp(const char *s1, const char *s2, size_t len)`

This function will compare two strings up to the given length.

#### Parametri

- **s1** -- pointer to the first string
- **s2** -- pointer to the second string
- **len** -- the maximum amount of characters to compare

#### Ritorna

the difference between the value of the first unmatched character.

`static inline bool lv_streq(const char *s1, const char *s2)`

Returns true if the two strings are equal. Just a wrapper around `strcmp` for convenience.

#### Parametri

- **s1** -- pointer to the first string
- **s2** -- pointer to the second string

#### Ritorna

true: the strings are equal; false: otherwise

`char *lv_strdup(const char *src)`

Duplicate a string by allocating a new one and copying the content.

#### Parametri

**src** -- Pointer to the source of data to be copied.

#### Ritorna

A pointer to the new allocated string. NULL if failed.

`char *lv_strndup(const char *src, size_t max_len)`

Duplicate a string by allocating a new one and copying the content up to the end or the specified maximum length, whichever comes first.

**Parametri**

- **src** -- Pointer to the source of data to be copied.
- **max\_len** -- Maximum number of characters to be copied.

**Ritorna**

Pointer to a newly allocated null-terminated string. NULL if failed.

`char *lv_strcat(char *dst, const char *src)`

Copies the string pointed to by src, including the terminating null character, to the end of the string pointed to by dst.

**Parametri**

- **dst** -- Pointer to the destination string where the content is to be appended.
- **src** -- Pointer to the source of data to be copied.

**Ritorna**

A pointer to the destination string, which is dst.

`char *lv_strncat(char *dst, const char *src, size_t src_len)`

Copies up to src\_len characters from the string pointed to by src to the end of the string pointed to by dst. A terminating null character is appended to dst even if no null character was encountered in src after src\_len characters were copied.

**Parametri**

- **dst** -- Pointer to the destination string where the content is to be appended.
- **src** -- Pointer to the source of data to be copied.
- **src\_len** -- Maximum number of characters from src to be copied to the end of dst.

**Ritorna**

A pointer to the destination string, which is dst.

`char *lv_strchr(const char *str, int c)`

Searches for the first occurrence of character c in the string str.

**Parametri**

- **str** -- Pointer to the null-terminated byte string to be searched.
- **c** -- The character to be searched for.

**Ritorna**

A pointer to the first occurrence of character c in the string str, or a null pointer if c is not found.

## 14.21.5 builtin

`lv_tlsf.h`

**Typedef**

`typedef void *lv_tlsf_t`

`typedef void *lv_pool_t`

`typedef void (*lv_tlsf_walker)(void *ptr, size_t size, int used, void *user)`

**Functions**

`lv_tlsf_t lv_tlsf_create(void *mem)`

`lv_tlsf_t lv_tlsf_create_with_pool(void *mem, size_t bytes)`

`void lv_tlsf_destroy(lv_tlsf_t tlsf)`

`lv_pool_t lv_tlsf_get_pool(lv_tlsf_t tlsf)`

```

lv_pool_t lv_tlsf_add_pool(lv_tlsf_t tlsf, void *mem, size_t bytes)
void lv_tlsf_remove_pool(lv_tlsf_t tlsf, lv_pool_t pool)
void *lv_tlsf_malloc(lv_tlsf_t tlsf, size_t bytes)
void *lv_tlsf_memalign(lv_tlsf_t tlsf, size_t align, size_t bytes)
void *lv_tlsf_realloc(lv_tlsf_t tlsf, void *ptr, size_t size)
size_t lv_tlsf_free(lv_tlsf_t tlsf, const void *ptr)
size_t lv_tlsf_block_size(void *ptr)
size_t lv_tlsf_size(void)
size_t lv_tlsf_align_size(void)
size_t lv_tlsf_block_size_min(void)
size_t lv_tlsf_block_size_max(void)
size_t lv_tlsf_pool_overhead(void)
size_t lv_tlsf_alloc_overhead(void)
void lv_tlsf_walk_pool(lv_pool_t pool, lv_tlsf_walker walker, void *user)
int lv_tlsf_check(lv_tlsf_t tlsf)
int lv_tlsf_check_pool(lv_pool_t pool)

```

## lv\_tlsf\_private.h

struct **lv\_tlsf\_state\_t**

### Public Members

*lv\_tlsf\_t* **tlsf**

size\_t **cur\_used**

size\_t **max\_used**

*lv\_ll\_t* **pool\_ll**

## 14.22 themes

### 14.22.1 lv\_theme.h

#### Typedef

typedef void (\***lv\_theme\_apply\_cb\_t**)(lv\_theme\_t\*, lv\_obj\_t\*)

## Functions

`lv_theme_t *lv_theme_get_from_obj(lv_obj_t *obj)`

Get the theme assigned to the display of the object

**Parametri**

**obj** -- pointer to a theme object

**Ritorna**

the theme of the object's display (can be NULL)

`void lv_theme_apply(lv_obj_t *obj)`

Apply the active theme on an object

**Parametri**

**obj** -- pointer to an object

`void lv_theme_set_parent(lv_theme_t *new_theme, lv_theme_t *parent)`

Set a base theme for a theme. The styles from the base them will be added before the styles of the current theme. Arbitrary long chain of themes can be created by setting base themes.

**Parametri**

- **new\_theme** -- pointer to theme which base should be set
- **parent** -- pointer to the base theme

`void lv_theme_set_apply_cb(lv_theme_t *theme, lv_theme_apply_cb_t apply_cb)`

Set an apply callback for a theme. The apply callback is used to add styles to different objects

**Parametri**

- **theme** -- pointer to theme which callback should be set
- **apply\_cb** -- pointer to the callback

`const lv_font_t *lv_theme_get_font_small(lv_obj_t *obj)`

Get the small font of the theme

**Parametri**

**obj** -- pointer to an object

**Ritorna**

pointer to the font

`const lv_font_t *lv_theme_get_font_normal(lv_obj_t *obj)`

Get the normal font of the theme

**Parametri**

**obj** -- pointer to an object

**Ritorna**

pointer to the font

`const lv_font_t *lv_theme_get_font_large(lv_obj_t *obj)`

Get the subtitle font of the theme

**Parametri**

**obj** -- pointer to an object

**Ritorna**

pointer to the font

`lv_color_t lv_theme_get_color_primary(lv_obj_t *obj)`

Get the primary color of the theme

**Parametri**

**obj** -- pointer to an object

**Ritorna**

the color

`lv_color_t lv_theme_get_color_secondary(lv_obj_t *obj)`

Get the secondary color of the theme

**Parametri**

**obj** -- pointer to an object

**Ritorna**

the color

## 14.22.2 lv\_theme\_private.h

struct **\_lv\_theme\_t**

### Public Members

*lv\_theme\_apply\_cb\_t* **apply\_cb**

**lv\_theme\_t \*parent**

Apply the current theme's style on top of this theme.

**void \*user\_data**

**lv\_display\_t \*disp**

*lv\_color\_t* **color\_primary**

*lv\_color\_t* **color\_secondary**

const **lv\_font\_t \*font\_small**

const **lv\_font\_t \*font\_normal**

const **lv\_font\_t \*font\_large**

**uint32\_t flags**

Any custom flag used by the theme

## 14.22.3 default

**lv\_theme\_default.h**

### Functions

**lv\_theme\_t \*lv\_theme\_default\_init(lv\_display\_t \*disp, *lv\_color\_t* color\_primary, *lv\_color\_t* color\_secondary, bool dark, const lv\_font\_t \*font)**

Initialize the theme

#### Parametri

- **disp** -- pointer to display
- **color\_primary** -- the primary color of the theme
- **color\_secondary** -- the secondary color for the theme
- **dark** --
- **font** -- pointer to a font to use.

#### Ritorna

a pointer to reference this theme later

**bool lv\_theme\_default\_is\_inited(void)**

Check if default theme is initialized

#### Ritorna

true if default theme is initialized, false otherwise

`lv_theme_t *lv_theme_default_get(void)`  
Get default theme  
**Ritorna**  
a pointer to default theme, or NULL if this is not initialized

`void lv_theme_default_deinit(void)`  
Deinitialize the default theme

#### 14.22.4 mono

`lv_theme_mono.h`

##### Functions

`lv_theme_t *lv_theme_mono_init(lv_display_t *disp, bool dark_bg, const lv_font_t *font)`

Initialize the theme

###### Parametri

- **disp** -- pointer to display
- **dark\_bg** --
- **font** -- pointer to a font to use.

###### Ritorna

a pointer to reference this theme later

`bool lv_theme_mono_is_inited(void)`

Check if the theme is initialized

###### Ritorna

true if default theme is initialized, false otherwise

`lv_theme_t *lv_theme_mono_get(void)`

Get mono theme

###### Ritorna

a pointer to mono theme, or NULL if this is not initialized

`void lv_theme_mono_deinit(void)`

Deinitialize the mono theme

#### 14.22.5 simple

`lv_theme_simple.h`

##### Functions

`lv_theme_t *lv_theme_simple_init(lv_display_t *disp)`

Initialize the theme

###### Parametri

- **disp** -- pointer to display

###### Ritorna

a pointer to reference this theme later

`bool lv_theme_simple_is_inited(void)`

Check if the theme is initialized

###### Ritorna

true if default theme is initialized, false otherwise

`lv_theme_t *lv_theme_simple_get(void)`

Get simple theme

###### Ritorna

a pointer to simple theme, or NULL if this is not initialized

`void lv_theme_simple_deinit(void)`

Deinitialize the simple theme

## 14.23 tick

### 14.23.1 lv\_tick.h

Provide access to the system tick with 1 millisecond resolution

#### Typedef

```
typedef uint32_t (*lv_tick_get_cb_t)(void)
```

```
typedef void (*lv_delay_cb_t)(uint32_t ms)
```

#### Functions

**void lv\_tick\_inc(uint32\_t tick\_period)**

You have to call this function periodically

##### Parametri

**tick\_period** -- the call period of this function in milliseconds

**uint32\_t lv\_tick\_get(void)**

Get the elapsed milliseconds since start up

##### Ritorna

the elapsed milliseconds

**uint32\_t lv\_tick\_elaps(uint32\_t prev\_tick)**

Get the elapsed milliseconds since a previous time stamp

##### Parametri

**prev\_tick** -- a previous time stamp (return value of [lv\\_tick\\_get\(\)](#))

##### Ritorna

the elapsed milliseconds since 'prev\_tick'

**uint32\_t lv\_tick\_diff(uint32\_t tick, uint32\_t prev\_tick)**

Get the elapsed milliseconds between two time stamps

##### Parametri

- **tick** -- a time stamp
- **prev\_tick** -- a time stamp before **tick**

##### Ritorna

the elapsed milliseconds between **prev\_tick** and **tick**

**void lv\_delay\_ms(uint32\_t ms)**

Delay for the given milliseconds. By default it's a blocking delay, but with [lv\\_delay\\_set\\_cb\(\)](#) a custom delay function can be set too

##### Parametri

**ms** -- the number of milliseconds to delay

**void lv\_delay\_set\_cb([lv\\_delay\\_cb\\_t](#) cb)**

Set a callback for a blocking delay

##### Parametri

**cb** -- pointer to a callback

**void lv\_tick\_set\_cb([lv\\_tick\\_get\\_cb\\_t](#) cb)**

Set the custom callback for 'lv\_tick\_get'

##### Parametri

**cb** -- call this callback on 'lv\_tick\_get'

[\*\*lv\\_tick\\_get\\_cb\\_t lv\\_tick\\_get\\_cb\(void\)\*\*](#)

Get the custom callback for 'lv\_tick\_get'

##### Ritorna

call this callback on 'lv\_tick\_get'

## 14.23.2 lv\_tick\_private.h

struct **lv\_tick\_state\_t**

### Public Members

**uint32\_t sys\_time**

**volatile uint8\_t sys\_irq\_flag**

*lv\_tick\_get\_cb\_t* **tick\_get\_cb**

*lv\_delay\_cb\_t* **delay\_cb**

## 14.24 widgets

### 14.24.1 3dtexture

**lv\_3dtexture.h**

#### Functions

**lv\_obj\_t \*lv\_3dtexture\_create(lv\_obj\_t \*parent)**

Create a 3dtexture object

##### Parametri

**parent** -- pointer to an object, it will be the parent of the new 3dtexture

##### Ritorna

pointer to the created 3dtexture

**void lv\_3dtexture\_set\_src(lv\_obj\_t \*obj, lv\_3dtexture\_id\_t id)**

Set the source texture of the widget. The object size should be manually set to match.

##### Parametri

- **obj** -- the 3dtexture widget
- **id** -- the texture handle from the 3D graphics backend. I.e., an `unsigned int` texture for OpenGL.

#### Variables

const lv\_obj\_class\_t **lv\_3dtexture\_class**

**lv\_3dtexture\_private.h**

struct **\_lv\_3dtexture\_t**

### Public Members

**lv\_obj\_t obj**

**lv\_3dtexture\_id\_t id**

## 14.24.2 animimage

**lv\_animimage.h**

### Enums

enum **lv\_property\_animimage\_id\_t**

Modified by NXP in 2025

*Values:*

enumerator **LV\_PROPERTY\_ANIMIMAGE\_SRC**

enumerator **LV\_PROPERTY\_ANIMIMAGE\_DURATION**

enumerator **LV\_PROPERTY\_ANIMIMAGE\_REPEAT\_COUNT**

enumerator **LV\_PROPERTY\_ANIMIMAGE\_SRC\_COUNT**

enumerator **LV\_PROPERTY\_ANIMIMAGE\_END**

enum **lv\_animimg\_part\_t**

Image parts

*Values:*

enumerator **LV\_ANIM\_IMAGE\_PART\_MAIN**

### Functions

**lv\_obj\_t \*lv\_animimg\_create(lv\_obj\_t \*parent)**

Create an animation image objects

#### Parametri

- **parent** -- pointer to an object, it will be the parent of the new button

#### Ritorna

- pointer to the created animation image object

**void lv\_animimg\_set\_src(lv\_obj\_t \*obj, const void \*dsc[], size\_t num)**

Set the image animation images source.

#### Parametri

- **obj** -- pointer to an animation image object
- **dsc** -- pointer to a series images
- **num** -- images' number

**void lv\_animimg\_set\_src\_reverse(lv\_obj\_t \*obj, const void \*dsc[], size\_t num)**

Set the images source for flip playback of animation image.

#### Parametri

- **obj** -- pointer to an animation image object
- **dsc** -- pointer to a series images
- **num** -- images' number

**void lv\_animimg\_start(lv\_obj\_t \*obj)**

Startup the image animation.

#### Parametri

- **obj** -- pointer to an animation image object

**bool lv\_animimg\_delete(lv\_obj\_t \*obj)**

Delete the image animation.

**Parametri**

- **obj** -- pointer to an animation image object

**void lv\_animimg\_set\_duration(lv\_obj\_t \*obj, uint32\_t duration)**

Set the image animation duration time. unit:ms

**Parametri**

- **obj** -- pointer to an animation image object
- **duration** -- the duration in milliseconds

**void lv\_animimg\_set\_repeat\_count(lv\_obj\_t \*obj, uint32\_t count)**

Set the image animation repeatedly play times.

**Parametri**

- **obj** -- pointer to an animation image object
- **count** -- the number of times to repeat the animation

**void lv\_animimg\_set\_reverse\_duration(lv\_obj\_t \*obj, uint32\_t duration)**

Make the image animation to play back to when the forward direction is ready.

**Parametri**

- **obj** -- pointer to an animation image object
- **duration** -- the duration of the playback image animation in milliseconds. 0: disable playback

**void lv\_animimg\_set\_reverse\_delay(lv\_obj\_t \*obj, uint32\_t duration)**

Make the image animation to play back to when the forward direction is ready.

**Parametri**

- **obj** -- pointer to an animation image object
- **duration** -- delay in milliseconds before starting the playback image animation.

**void lv\_animimg\_set\_start\_cb(lv\_obj\_t \*obj, lv\_anim\_start\_cb\_t start\_cb)**

Set a function call when the animation image really starts (considering **delay**)

**Parametri**

- **obj** -- pointer to an animation image object
- **start\_cb** -- a function call when the animation is start

**void lv\_animimg\_set\_completed\_cb(lv\_obj\_t \*obj, lv\_anim\_completed\_cb\_t completed\_cb)**

Set a function call when the animation is completed

**Parametri**

- **obj** -- pointer to an animation image object
- **completed\_cb** -- a function call when the animation is completed

**const void \*\*lv\_animimg\_get\_src(lv\_obj\_t \*obj)**

Get the image animation images source.

**Parametri**

- **obj** -- pointer to an animation image object

**Ritorna**

a pointer that will point to a series images

**uint8\_t lv\_animimg\_get\_src\_count(lv\_obj\_t \*obj)**

Get the image animation images source.

**Parametri**

- **obj** -- pointer to an animation image object

**Ritorna**

the number of source images

**uint32\_t lv\_animimg\_get\_duration(lv\_obj\_t \*obj)**

Get the image animation duration time. unit:ms

**Parametri**

- **obj** -- pointer to an animation image object

**Ritorna**

the animation duration time

```
uint32_t lv_animimg_get_repeat_count(lv_obj_t *obj)
```

Get the image animation repeat play times.

**Parametri**

**obj** -- pointer to an animation image object

**Ritorna**

the repeat count

```
lv_anim_t *lv_animimg_get_anim(lv_obj_t *obj)
```

Get the image animation underlying animation.

**Parametri**

**obj** -- pointer to an animation image object

**Ritorna**

the animation reference

**Variables**

```
const lv_obj_class_t lv_animimg_class
```

**lv\_animimage\_private.h**

```
struct _lv_animimg_t
#include <lv_animimage_private.h> Data of the animimage
```

**Public Members**

lv\_image\_t **img**

lv\_anim\_t **anim**

const void \*\***dsc**

int8\_t **pic\_count**

**14.24.3 arc****lv\_arc.h****Enums**

```
enum lv_arc_mode_t
```

*Values:*

enumerator **LV\_ARC\_MODE\_NORMAL**

enumerator **LV\_ARC\_MODE\_SYMMETRICAL**

enumerator **LV\_ARC\_MODE\_REVERSE**

## Functions

`lv_obj_t *lv_arc_create(lv_obj_t *parent)`

Create an arc object

### Parametri

- **parent** -- pointer to an object, it will be the parent of the new arc

### Ritorna

- pointer to the created arc

`void lv_arc_set_start_angle(lv_obj_t *obj, lv_value_precise_t start)`

Set the start angle of an arc. 0 deg: right, 90 bottom, etc.

### Parametri

- **obj** -- pointer to an arc object
- **start** -- the start angle. (if `LV_USE_FLOAT` is enabled it can be fractional too.)

`void lv_arc_set_end_angle(lv_obj_t *obj, lv_value_precise_t end)`

Set the end angle of an arc. 0 deg: right, 90 bottom, etc.

### Parametri

- **obj** -- pointer to an arc object
- **end** -- the end angle (if `LV_USE_FLOAT` is enabled it can be fractional too.)

`void lv_arc_set_angles(lv_obj_t *obj, lv_value_precise_t start, lv_value_precise_t end)`

Set the start and end angles

### Parametri

- **obj** -- pointer to an arc object
- **start** -- the start angle (if `LV_USE_FLOAT` is enabled it can be fractional too.)
- **end** -- the end angle (if `LV_USE_FLOAT` is enabled it can be fractional too.)

`void lv_arc_set_bg_start_angle(lv_obj_t *obj, lv_value_precise_t start)`

Set the start angle of an arc background. 0 deg: right, 90 bottom, etc.

### Parametri

- **obj** -- pointer to an arc object
- **start** -- the start angle (if `LV_USE_FLOAT` is enabled it can be fractional too.)

`void lv_arc_set_bg_end_angle(lv_obj_t *obj, lv_value_precise_t end)`

Set the start angle of an arc background. 0 deg: right, 90 bottom etc.

### Parametri

- **obj** -- pointer to an arc object
- **end** -- the end angle (if `LV_USE_FLOAT` is enabled it can be fractional too.)

`void lv_arc_set_bg_angles(lv_obj_t *obj, lv_value_precise_t start, lv_value_precise_t end)`

Set the start and end angles of the arc background

### Parametri

- **obj** -- pointer to an arc object
- **start** -- the start angle (if `LV_USE_FLOAT` is enabled it can be fractional too.)
- **end** -- the end angle (if `LV_USE_FLOAT` is enabled it can be fractional too.)

`void lv_arc_set_rotation(lv_obj_t *obj, int32_t rotation)`

Set the rotation for the whole arc

### Parametri

- **obj** -- pointer to an arc object
- **rotation** -- rotation angle

`void lv_arc_set_mode(lv_obj_t *obj, lv_arc_mode_t type)`

Set the type of arc.

### Parametri

- **obj** -- pointer to arc object
- **type** -- arc's mode

`void lv_arc_set_value(lv_obj_t *obj, int32_t value)`

Set a new value on the arc

**Parametri**

- **obj** -- pointer to an arc object
- **value** -- new value

**void lv\_arc\_set\_range(lv\_obj\_t \*obj, int32\_t min, int32\_t max)**

Set minimum and the maximum values of an arc

**Parametri**

- **obj** -- pointer to the arc object
- **min** -- minimum value
- **max** -- maximum value

**void lv\_arc\_set\_min\_value(lv\_obj\_t \*obj, int32\_t min)**

Set the minimum values of an arc

**Parametri**

- **obj** -- pointer to the arc object
- **min** -- minimum value

**void lv\_arc\_set\_max\_value(lv\_obj\_t \*obj, int32\_t max)**

Set the maximum values of an arc

**Parametri**

- **obj** -- pointer to the arc object
- **max** -- maximum value

**void lv\_arc\_set\_change\_rate(lv\_obj\_t \*obj, uint32\_t rate)**

Set a change rate to limit the speed how fast the arc should reach the pressed point.

**Parametri**

- **obj** -- pointer to an arc object
- **rate** -- the change rate

**void lv\_arc\_set\_knob\_offset(lv\_obj\_t \*obj, int32\_t offset)**

Set an offset angle for the knob

**Parametri**

- **obj** -- pointer to an arc object
- **offset** -- knob offset from main arc in degrees

**lv\_value\_precise\_t lv\_arc\_get\_angle\_start(lv\_obj\_t \*obj)**

Get the start angle of an arc.

**Parametri**

**obj** -- pointer to an arc object

**Ritorna**

the start angle [0..360] (if LV\_USE\_FLOAT is enabled it can be fractional too.)

**lv\_value\_precise\_t lv\_arc\_get\_angle\_end(lv\_obj\_t \*obj)**

Get the end angle of an arc.

**Parametri**

**obj** -- pointer to an arc object

**Ritorna**

the end angle [0..360] (if LV\_USE\_FLOAT is enabled it can be fractional too.)

**lv\_value\_precise\_t lv\_arc\_get\_bg\_angle\_start(lv\_obj\_t \*obj)**

Get the start angle of an arc background.

**Parametri**

**obj** -- pointer to an arc object

**Ritorna**

the start angle [0..360] (if LV\_USE\_FLOAT is enabled it can be fractional too.)

**lv\_value\_precise\_t lv\_arc\_get\_bg\_angle\_end(lv\_obj\_t \*obj)**

Get the end angle of an arc background.

**Parametri**

**obj** -- pointer to an arc object

**Ritorna**

the end angle [0..360] (if LV\_USE\_FLOAT is enabled it can be fractional too.)

`int32_t lv_arc_get_value(const lv_obj_t *obj)`

Get the value of an arc

**Parametri**

**obj** -- pointer to an arc object

**Ritorna**

the value of the arc

`int32_t lv_arc_get_min_value(const lv_obj_t *obj)`

Get the minimum value of an arc

**Parametri**

**obj** -- pointer to an arc object

**Ritorna**

the minimum value of the arc

`int32_t lv_arc_get_max_value(const lv_obj_t *obj)`

Get the maximum value of an arc

**Parametri**

**obj** -- pointer to an arc object

**Ritorna**

the maximum value of the arc

`lv_arc_mode_t lv_arc_get_mode(const lv_obj_t *obj)`

Get whether the arc is type or not.

**Parametri**

**obj** -- pointer to an arc object

**Ritorna**

arc's mode

`int32_t lv_arc_get_rotation(const lv_obj_t *obj)`

Get the rotation for the whole arc

**Parametri**

**obj** -- pointer to an arc object

**Ritorna**

arc's current rotation

`int32_t lv_arc_get_knob_offset(const lv_obj_t *obj)`

Get the current knob angle offset

**Parametri**

**obj** -- pointer to an arc object

**Ritorna**

arc's current knob offset

`void lv_arc_align_obj_to_angle(const lv_obj_t *obj, lv_obj_t *obj_to_align, int32_t r_offset)`

Align an object to the current position of the arc (knob)

**Parametri**

- **obj** -- pointer to an arc object
- **obj\_to\_align** -- pointer to an object to align
- **r\_offset** -- consider the radius larger with this value (< 0: for smaller radius)

`void lv_arc_rotate_obj_to_angle(const lv_obj_t *obj, lv_obj_t *obj_to_rotate, int32_t r_offset)`

Rotate an object to the current position of the arc (knob)

**Parametri**

- **obj** -- pointer to an arc object
- **obj\_to\_rotate** -- pointer to an object to rotate
- **r\_offset** -- consider the radius larger with this value (< 0: for smaller radius)

## Variables

const lv\_obj\_class\_t **lv\_arc\_class**

### lv\_arc\_private.h

struct **\_lv\_arc\_t**

#### Public Members

lv\_obj\_t **obj**

int32\_t **rotation**

lv\_value\_precise\_t **indic\_angle\_start**

lv\_value\_precise\_t **indic\_angle\_end**

lv\_value\_precise\_t **bg\_angle\_start**

lv\_value\_precise\_t **bg\_angle\_end**

int32\_t **value**

Current value of the arc

int32\_t **min\_value**

Minimum value of the arc

int32\_t **max\_value**

Maximum value of the arc

uint32\_t **dragging**

uint32\_t **type**

uint32\_t **min\_close**

1: the last pressed angle was closer to minimum end

uint32\_t **in\_out**

1: The click was within the background arc angles. 0: Click outside

uint32\_t **chg\_rate**

Drag angle rate of change of the arc (degrees/sec)

uint32\_t **last\_tick**

Last dragging event timestamp of the arc

`lv_value_precise_t last_angle`

Last dragging angle of the arc

`int16_t knob_offset`

knob offset from the main arc

#### 14.24.4 arclabel

`lv_arclabel.h`

**Defines**

`LV_ARCLABEL_DOT_NUM`

`LV_ARCLABEL_DEFAULT_TEXT`

**Enums**

`enum lv_arclabel_dir_t`

*Values:*

enumerator `LV_ARCLABEL_DIR_CLOCKWISE`

enumerator `LV_ARCLABEL_DIR_COUNTER_CLOCKWISE`

`enum lv_arclabel_text_align_t`

*Values:*

enumerator `LV_ARCLABEL_TEXT_ALIGN_DEFAULT`

enumerator `LV_ARCLABEL_TEXT_ALIGN.LEADING`

enumerator `LV_ARCLABEL_TEXT_ALIGN.CENTER`

enumerator `LV_ARCLABEL_TEXT_ALIGN.TRAILING`

**Functions**

`lv_obj_t *lv_arclabel_create(lv_obj_t *parent)`

Create an arc label object

**Parametri**

**parent** -- pointer to an object, it will be the parent of the new arc label

**Ritorna**

pointer to the created arc label

`void lv_arclabel_set_text(lv_obj_t *obj, const char *text)`

Set the text of the arc label.

This function sets the text displayed by an arc label object.

**Parametri**

- **obj** -- Pointer to the arc label object.
- **text** -- Pointer to a null-terminated string containing the new text for the label.

---

```
void lv_arclabel_set_text_fmt(lv_obj_t *obj, const char *fmt, ...)
```

Set the formatted text of an arc label object.

This function sets the text of an arc label object with support for variable arguments formatting, similar to `printf`.

**Parametri**

- **obj** -- The arc label object to set the text for.
- **fmt** -- A format string that specifies how subsequent arguments are converted to text.
- **...** -- Arguments following the format string that are used to replace format specifiers in the format string.

```
void lv_arclabel_set_text_static(lv_obj_t *obj, const char *text)
```

Sets a new static text for the arc label or refreshes it with the current text. The 'text' must remain valid in memory; the arc label does not manage its lifecycle.

**Parametri**

- **obj** -- Pointer to the arc label object.
- **text** -- Pointer to the new text. If NULL, the label is refreshed with its current text.

```
void lv_arclabel_set_angle_start(lv_obj_t *obj, lv_value_precise_t start)
```

Set the start angle of an arc. 0 deg: right, 90 bottom, etc.

**Parametri**

- **obj** -- pointer to an arc label object
- **start** -- the start angle. (if `LV_USE_FLOAT` is enabled it can be fractional too.)

```
void lv_arclabel_set_angle_size(lv_obj_t *obj, lv_value_precise_t size)
```

Set the end angle of an arc. 0 deg: right, 90 bottom, etc.

**Parametri**

- **obj** -- pointer to an arc label object
- **size** -- the angle size (if `LV_USE_FLOAT` is enabled it can be fractional too.)

```
void lv_arclabel_set_offset(lv_obj_t *obj, int32_t offset)
```

Set the rotation for the whole arc

**Parametri**

- **obj** -- pointer to an arc label object
- **offset** -- rotation angle

```
void lv_arclabel_set_dir(lv_obj_t *obj, lv_arclabel_dir_t dir)
```

Set the type of arc.

**Parametri**

- **obj** -- pointer to an arc label object
- **dir** -- arc label's direction

```
void lv_arclabel_set_recolor(lv_obj_t *obj, bool en)
```

Enable the recoloring by in-line commands

**Parametri**

- **obj** -- pointer to an arc label object
- **en** -- true: enable recoloring, false: disable Example: "This is a #ff0000 red# word"

```
void lv_arclabel_set_radius(lv_obj_t *obj, uint32_t radius)
```

Set the radius for an arc label object.

**Parametri**

- **obj** -- pointer to the arc label object.
- **radius** -- The radius value to set for the label's curvature, in pixels.

```
void lv_arclabel_set_center_offset_x(lv_obj_t *obj, uint32_t x)
```

Set the center offset x for an arc label object.

**Parametri**

- **obj** -- pointer to an arc label object
- **x** -- the x offset

**void lv\_arclabel\_set\_center\_offset\_y(lv\_obj\_t \*obj, uint32\_t y)**

Set the center offset y for an arc label object.

**Parametri**

- **obj** -- pointer to an arc label object
- **y** -- the y offset

**void lv\_arclabel\_set\_text\_vertical\_align(lv\_obj\_t \*obj, lv\_arclabel\_text\_align\_t align)**

Set the text vertical alignment for an arc label object.

**Parametri**

- **obj** -- pointer to an arc label object
- **align** -- the vertical alignment

**void lv\_arclabel\_set\_text\_horizontal\_align(lv\_obj\_t \*obj, lv\_arclabel\_text\_align\_t align)**

Set the text horizontal alignment for an arc label object.

**Parametri**

- **obj** -- pointer to an arc label object
- **align** -- the horizontal alignment

**lv\_value\_precise\_t lv\_arclabel\_get\_angle\_start(lv\_obj\_t \*obj)**

Get the start angle of an arc label.

**Parametri**

- **obj** -- pointer to an arc label object

**Ritorna**

the start angle [0..360] (if LV\_USE\_FLOAT is enabled it can be fractional too.)

**lv\_value\_precise\_t lv\_arclabel\_get\_angle\_size(lv\_obj\_t \*obj)**

Get the angle size of an arc label.

**Parametri**

- **obj** -- pointer to an arc label object

**Ritorna**

the end angle [0..360] (if LV\_USE\_FLOAT is enabled it can be fractional too.)

**lv\_arclabel\_dir\_t lv\_arclabel\_get\_dir(const lv\_obj\_t \*obj)**

Get whether the arc label is type or not.

**Parametri**

- **obj** -- pointer to an arc label object

**Ritorna**

arc label's direction

**bool lv\_arclabel\_get\_recolor(lv\_obj\_t \*obj)**

Enable the recoloring by in-line commands

 **Vedi anche**

[lv\\_arclabel\\_set\\_recolor](#)

**Parametri**

- **obj** -- pointer to a label object

**Ritorna**

true: enable recoloring, false: disable

**uint32\_t lv\_arclabel\_get\_radius(lv\_obj\_t \*obj)**

Get the text of the arc label.

**Parametri**

- **obj** -- pointer to an arc label object

**Ritorna**

the radius of the arc label

**uint32\_t lv\_arclabel\_get\_center\_offset\_x(lv\_obj\_t \*obj)**

Get the center offset x for an arc label object.

**Parametri**

**obj** -- pointer to an arc label object

**Ritorna**

the x offset

`uint32_t lv_arclabel_get_center_offset_x(lv_obj_t *obj)`

Get the center offset x for an arc label object.

**Parametri**

**obj** -- pointer to an arc label object

**Ritorna**

the y offset

`lv_arclabel_text_align_t lv_arclabel_get_text_vertical_align(lv_obj_t *obj)`

Get the text vertical alignment for an arc label object.

**Parametri**

**obj** -- pointer to an arc label object

**Ritorna**

the vertical alignment

`lv_arclabel_text_align_t lv_arclabel_get_text_horizontal_align(lv_obj_t *obj)`

Get the text horizontal alignment for an arc label object.

**Parametri**

**obj** -- pointer to an arc label object

**Ritorna**

the horizontal alignment

**Variables**

`const lv_obj_class_t lv_arclabel_class`

**lv\_arclabel\_private.h**

`struct _lv_arclabel_t`

**Public Members**

`lv_obj_t obj`

`char *text`

`char dot[3 + 1]`

Bytes that have been replaced with dots

`uint32_t dot_begin`

Offset where bytes have been replaced with dots

`lv_value_precise_t angle_start`

The starting angle of the arc in degrees.

This variable represents the beginning of the arc's angular range in terms of degrees. It is used in conjunction with `angle_size` to define the section of the circle that the arc covers. Values are normalized to the range [0, 360), ensuring compatibility with circular representations.

Modifying this value affects the visual rendering of the arc and its associated indicators or labels, necessitating a subsequent invalidation or update of the object's display to reflect the changes.

**Nota**

When setting this value programmatically, ensure it does not exceed 360 degrees, as it will be automatically adjusted to fit within the valid range.

`lv_value_precise_t angle_size`

`uint32_t offset`

`uint32_t radius`

`lv_point_t center_offset`

`lv_arclabel_dir_t dir`

`lv_arclabel_text_align_t text_align_v`

Vertical text alignment

`lv_arclabel_text_align_t text_align_h`

Horizontal text alignment

`uint8_t static_txt`

Flag to indicate the text is static

`uint8_t recolor`

Enable in-line letter re-coloring

## 14.24.5 bar

`lv_bar.h`

**Enums**

enum `lv_bar_mode_t`

*Values:*

enumerator `LV_BAR_MODE_NORMAL`

enumerator `LV_BAR_MODE_SYMMETRICAL`

enumerator `LV_BAR_MODE_RANGE`

enum `lv_bar_orientation_t`

*Values:*

enumerator `LV_BAR_ORIENTATION_AUTO`

enumerator `LV_BAR_ORIENTATION_HORIZONTAL`

enumerator **LV\_BAR\_ORIENTATION\_VERTICAL**

## Functions

`lv_obj_t *lv_bar_create(lv_obj_t *parent)`

Create a bar object

### Parametri

**parent** -- pointer to an object, it will be the parent of the new bar

### Ritorna

pointer to the created bar

`void lv_bar_set_value(lv_obj_t *obj, int32_t value, lv_anim_enable_t anim)`

Set a new value on the bar

### Parametri

- **obj** -- pointer to a bar object
- **value** -- new value
- **anim** -- LV\_ANIM\_ON: set the value with an animation; LV\_ANIM\_OFF: change the value immediately

`void lv_bar_set_start_value(lv_obj_t *obj, int32_t start_value, lv_anim_enable_t anim)`

Set a new start value on the bar

### Parametri

- **obj** -- pointer to a bar object
- **start\_value** -- new start value
- **anim** -- LV\_ANIM\_ON: set the value with an animation; LV\_ANIM\_OFF: change the value immediately

`void lv_bar_set_range(lv_obj_t *obj, int32_t min, int32_t max)`

Set minimum and the maximum values of a bar

### Nota

If min is greater than max, the drawing direction becomes to the opposite direction.

### Parametri

- **obj** -- pointer to the bar object
- **min** -- minimum value
- **max** -- maximum value

`void lv_bar_set_min_value(lv_obj_t *obj, int32_t min)`

Set minimum value of a bar

### Parametri

- **obj** -- pointer to the bar object
- **min** -- minimum value

`void lv_bar_set_max_value(lv_obj_t *obj, int32_t max)`

Set maximum value of a bar

### Parametri

- **obj** -- pointer to the bar object
- **max** -- maximum value

`void lv_bar_set_mode(lv_obj_t *obj, lv_bar_mode_t mode)`

Set the type of bar.

### Parametri

- **obj** -- pointer to bar object
- **mode** -- bar type from `lv_bar_mode_t`

`void lv_bar_set_orientation(lv_obj_t *obj, lv_bar_orientation_t orientation)`

Set the orientation of bar.

**Parametri**

- **obj** -- pointer to bar object
- **orientation** -- bar orientation from `lv_bar_orientation_t`

`int32_t lv_bar_get_value(const lv_obj_t *obj)`

Get the value of a bar

**Parametri**

- **obj** -- pointer to a bar object

**Ritorna**

the value of the bar

`int32_t lv_bar_get_start_value(const lv_obj_t *obj)`

Get the start value of a bar

**Parametri**

- **obj** -- pointer to a bar object

**Ritorna**

the start value of the bar

`int32_t lv_bar_get_min_value(const lv_obj_t *obj)`

Get the minimum value of a bar

**Parametri**

- **obj** -- pointer to a bar object

**Ritorna**

the minimum value of the bar

`int32_t lv_bar_get_max_value(const lv_obj_t *obj)`

Get the maximum value of a bar

**Parametri**

- **obj** -- pointer to a bar object

**Ritorna**

the maximum value of the bar

`lv_bar_mode_t lv_bar_get_mode(lv_obj_t *obj)`

Get the type of bar.

**Parametri**

- **obj** -- pointer to bar object

**Ritorna**

bar type from `lv_bar_mode_t`

`lv_bar_orientation_t lv_bar_get_orientation(lv_obj_t *obj)`

Get the orientation of bar.

**Parametri**

- **obj** -- pointer to bar object

**Ritorna**

bar orientation from `lv_bar_orientation_t`

`bool lv_bar_is_symmetrical(lv_obj_t *obj)`

Give the bar is in symmetrical mode or not

**Parametri**

- **obj** -- pointer to bar object

**Ritorna**

true: in symmetrical mode false : not in

## Variables

const lv\_obj\_class\_t **lv\_bar\_class**

**lv\_bar\_private.h**

struct **\_lv\_bar\_anim\_t**

### Public Members

lv\_obj\_t \***bar**

int32\_t **anim\_start**

int32\_t **anim\_end**

int32\_t **anim\_state**

struct **\_lv\_bar\_t**

### Public Members

lv\_obj\_t **obj**

int32\_t **cur\_value**

Current value of the bar

int32\_t **min\_value**

Minimum value of the bar

int32\_t **max\_value**

Maximum value of the bar

int32\_t **start\_value**

Start value of the bar

*lv\_area\_t* **indic\_area**

Save the indicator area. Might be used by derived types

bool **val\_reversed**

Whether value been reversed

*lv\_bar\_anim\_t* **cur\_value\_anim**

*lv\_bar\_anim\_t* **start\_value\_anim**

*lv\_bar\_mode\_t* **mode**

Type of bar

***lv\_bar\_orientation\_t*** **orientation**

Orientation of bar

**14.24.6 button****lv\_button.h****Functions**

`lv_obj_t *lv_button_create(lv_obj_t *parent)`

Create a button object

**Parametri**

**parent** -- pointer to an object, it will be the parent of the new button

**Ritorna**

pointer to the created button

**Variables**

`const lv_obj_class_t lv_button_class`

**lv\_button\_private.h**

`struct _lv_button_t`

**Public Members**

`lv_obj_t obj`

**14.24.7 buttonmatrix****lv\_buttonmatrix.h****Defines**

**LV\_BUTTONMATRIX\_BUTTON\_NONE**

**Typedef**

`typedef bool (*lv_buttonmatrix_button_draw_cb_t)(lv_obj_t *btm, uint32_t btn_id, const lv_area_t *draw_area, const lv_area_t *clip_area)`

**Enums**

`enum lv_buttonmatrix_ctrl_t`

Type to store button control flags (disabled, hidden etc.) The least-significant 4 bits are used to store button-width proportions in range [1..15].

*Values:*

enumerator **LV\_BUTTONMATRIX\_CTRL\_NONE**

No extra control, use the default settings

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_1`**

Set the width to 1 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_2`**

Set the width to 2 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_3`**

Set the width to 3 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_4`**

Set the width to 4 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_5`**

Set the width to 5 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_6`**

Set the width to 6 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_7`**

Set the width to 7 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_8`**

Set the width to 8 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_9`**

Set the width to 9 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_10`**

Set the width to 10 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_11`**

Set the width to 11 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_12`**

Set the width to 12 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_13`**

Set the width to 13 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_14`**

Set the width to 14 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_WIDTH_15`**

Set the width to 15 relative to the other buttons in the same row

**enumerator `LV_BUTTONMATRIX_CTRL_HIDDEN`**

Hides button; it continues to hold its space in layout.

**enumerator LV\_BUTTONMATRIX\_CTRL\_NO\_REPEAT**

Do not emit LV\_EVENT\_LONG\_PRESSED\_REPEAT events while button is long-pressed.

**enumerator LV\_BUTTONMATRIX\_CTRL\_DISABLED**

Disables button like LV\_STATE\_DISABLED on normal Widgets.

**enumerator LV\_BUTTONMATRIX\_CTRL\_CHECKABLE**

Enable toggling of LV\_STATE\_CHECKED when clicked.

**enumerator LV\_BUTTONMATRIX\_CTRL\_CHECKED**

Make the button checked. It will use the :cpp:enumerator:LV\_STATE\_CHECHKED styles.

**enumerator LV\_BUTTONMATRIX\_CTRL\_CLICK\_TRIG**

1: Enables sending LV\_EVENT\_VALUE\_CHANGE on CLICK, 0: sends LV\_EVENT\_VALUE\_CHANGE on PRESS.

**enumerator LV\_BUTTONMATRIX\_CTRL\_POPOVER**

Show button text in a pop-over while being pressed.

**enumerator LV\_BUTTONMATRIX\_CTRL\_RECOLOR**

Enable text recoloring with #color

**enumerator LV\_BUTTONMATRIX\_CTRL\_RESERVED\_1**

Reserved for later use

**enumerator LV\_BUTTONMATRIX\_CTRL\_RESERVED\_2**

Reserved for later use

**enumerator LV\_BUTTONMATRIX\_CTRL\_CUSTOM\_1**

Custom free-to-use flag

**enumerator LV\_BUTTONMATRIX\_CTRL\_CUSTOM\_2**

Custom free-to-use flag

**Functions**

`lv_obj_t *lv_buttonmatrix_create(lv_obj_t *parent)`

Create a button matrix object

**Parametri**

- **parent** -- pointer to an object, it will be the parent of the new button matrix

**Ritorna**

pointer to the created button matrix

`void lv_buttonmatrix_set_map(lv_obj_t *obj, const char *const map[])`

Set a new map. Buttons will be created/deleted according to the map. The button matrix keeps a reference to the map and so the string array must not be deallocated during the life of the matrix.

**Parametri**

- **obj** -- pointer to a button matrix object
- **map** -- pointer a string array. The last string has to be "". Use "\n" to make a line break.

---

**void lv\_buttonmatrix\_set\_ctrl\_map(lv\_obj\_t \*obj, const *lv\_buttonmatrix\_ctrl\_t* ctrl\_map[])**

Set the button control map (hidden, disabled etc.) for a button matrix. The control map array will be copied and so may be deallocated after this function returns.

**Parametri**

- **obj** -- pointer to a button matrix object
- **ctrl\_map** -- pointer to an array of *lv\_button\_ctrl\_t* control bytes. The length of the array and position of the elements must match the number and order of the individual buttons (i.e. excludes newline entries). An element of the map should look like e.g.: `ctrl_map[0] = width | LV_BUTTONMATRIX_CTRL_NO_REPEAT | LV_BUTTONMATRIX_CTRL_TGL_ENABLE`

**void lv\_buttonmatrix\_set\_selected\_button(lv\_obj\_t \*obj, uint32\_t btn\_id)**

Set the selected buttons

**Parametri**

- **obj** -- pointer to button matrix object
- **btn\_id** -- 0 based index of the button to modify. (Not counting new lines)

**void lv\_buttonmatrix\_set\_button\_ctrl(lv\_obj\_t \*obj, uint32\_t btn\_id, *lv\_buttonmatrix\_ctrl\_t* ctrl)**

Set the attributes of a button of the button matrix

**Parametri**

- **obj** -- pointer to button matrix object
- **btn\_id** -- 0 based index of the button to modify. (Not counting new lines)
- **ctrl** -- OR-ed attributes. E.g. `LV_BUTTONMATRIX_CTRL_NO_REPEAT | LV_BUTTONMATRIX_CTRL_CHECKABLE`

**void lv\_buttonmatrix\_clear\_button\_ctrl(lv\_obj\_t \*obj, uint32\_t btn\_id, *lv\_buttonmatrix\_ctrl\_t* ctrl)**

Clear the attributes of a button of the button matrix

**Parametri**

- **obj** -- pointer to button matrix object
- **btn\_id** -- 0 based index of the button to modify. (Not counting new lines)
- **ctrl** -- OR-ed attributes. E.g. `LV_BUTTONMATRIX_CTRL_NO_REPEAT | LV_BUTTONMATRIX_CTRL_CHECKABLE`

**void lv\_buttonmatrix\_set\_button\_ctrl\_all(lv\_obj\_t \*obj, *lv\_buttonmatrix\_ctrl\_t* ctrl)**

Set attributes of all buttons of a button matrix

**Parametri**

- **obj** -- pointer to a button matrix object
- **ctrl** -- attribute(s) to set from *lv\_buttonmatrix\_ctrl\_t*. Values can be ORed.

**void lv\_buttonmatrix\_clear\_button\_ctrl\_all(lv\_obj\_t \*obj, *lv\_buttonmatrix\_ctrl\_t* ctrl)**

Clear the attributes of all buttons of a button matrix

**Parametri**

- **obj** -- pointer to a button matrix object
- **ctrl** -- attribute(s) to set from *lv\_buttonmatrix\_ctrl\_t*. Values can be ORed.

**void lv\_buttonmatrix\_set\_button\_width(lv\_obj\_t \*obj, uint32\_t btn\_id, uint32\_t width)**

Set a single button's relative width. This method will cause the matrix be regenerated and is a relatively expensive operation. It is recommended that initial width be specified using *lv\_buttonmatrix\_set\_ctrl\_map* and this method only be used for dynamic changes.

**Parametri**

- **obj** -- pointer to button matrix object
- **btn\_id** -- 0 based index of the button to modify.
- **width** -- relative width compared to the buttons in the same row. [1..15]

**void lv\_buttonmatrix\_set\_one\_checked(lv\_obj\_t \*obj, bool en)**

Make the button matrix like a selector widget (only one button may be checked at a time).

`LV_BUTTONMATRIX_CTRL_CHECKABLE` must be enabled on the buttons to be selected using `lv_buttonmatrix_set_ctrl()` or `lv_buttonmatrix_set_button_ctrl_all()`.

**Parametri**

- **obj** -- pointer to a button matrix object
- **en** -- whether "one check" mode is enabled

`const char *const *lv_buttonmatrix_get_map(const lv_obj_t *obj)`

Get the current map of a button matrix

**Parametri**

- **obj** -- pointer to a button matrix object

**Ritorna**

the current map

`uint32_t lv_buttonmatrix_get_selected_button(const lv_obj_t *obj)`

Get the index of the lastly "activated" button by the user (pressed, released, focused etc) Useful in the `event_cb` to get the text of the button, check if hidden etc.

**Parametri**

- **obj** -- pointer to button matrix object

**Ritorna**

index of the last released button (`LV_BUTTONMATRIX_BUTTON_NONE`: if unset)

`const char *lv_buttonmatrix_get_button_text(const lv_obj_t *obj, uint32_t btn_id)`

Get the button's text

**Parametri**

- **obj** -- pointer to button matrix object
- **btn\_id** -- the index a button not counting new line characters.

**Ritorna**

text of `btn_index`` button

`bool lv_buttonmatrix_has_button_ctrl(lv_obj_t *obj, uint32_t btn_id, lv_buttonmatrix_ctrl_t ctrl)`

Get the whether a control value is enabled or disabled for button of a button matrix

**Parametri**

- **obj** -- pointer to a button matrix object
- **btn\_id** -- the index of a button not counting new line characters.
- **ctrl** -- control values to check (ORed value can be used)

**Ritorna**

true: the control attribute is enabled false: disabled

`bool lv_buttonmatrix_get_one_checked(const lv_obj_t *obj)`

Tell whether "one check" mode is enabled or not.

**Parametri**

- **obj** -- Button matrix object

**Ritorna**

true: "one check" mode is enabled; false: disabled

## Variables

`const lv_obj_class_t lv_buttonmatrix_class`

`lv_buttonmatrix_private.h`

`struct _lv_buttonmatrix_t`

`#include <lv_buttonmatrix_private.h>` Data of button matrix

**Public Members****lv\_obj\_t obj****const char \*const \*map\_p**

Pointer to the current map

**lv\_area\_t \*button\_areas**

Array of areas of buttons

**lv\_buttonmatrix\_ctrl\_t \*ctrl\_bits**

Array of control bytes

**uint32\_t btn\_cnt**

Number of button in 'map\_p'(Handled by the library)

**uint32\_t row\_cnt**

Number of rows in 'map\_p'(Handled by the library)

**uint32\_t btn\_id\_sel**

Index of the active button (being pressed/released etc) or LV\_BUTTONMATRIX\_BUTTON\_NONE

**uint32\_t one\_check**

1: Single button toggled at once

**uint32\_t auto\_free\_map**

1: Automatically free the map when the widget is deleted

## 14.24.8 calendar

**lv\_calendar.h****Functions****lv\_obj\_t \*lv\_calendar\_create(lv\_obj\_t \*parent)**

Create a calendar widget

**Parametri**

parent -- pointer to an object, it will be the parent of the new calendar

**Ritorna**

pointer the created calendar

**void lv\_calendar\_set\_today\_date(lv\_obj\_t \*obj, uint32\_t year, uint32\_t month, uint32\_t day)**

Set the today's year, month and day at once

**Parametri**

- **obj** -- pointer to a calendar object
- **year** -- today's year
- **month** -- today's month [1..12]
- **day** -- today's day [1..31]

**void lv\_calendar\_set\_today\_year(lv\_obj\_t \*obj, uint32\_t year)**

Set the today's year

**Parametri**

- **obj** -- pointer to a calendar object
- **year** -- today's year

**void lv\_calendar\_set\_today\_month(lv\_obj\_t \*obj, uint32\_t month)**

Set the today's year

**Parametri**

- **obj** -- pointer to a calendar object
- **month** -- today's month [1..12]

**void lv\_calendar\_set\_today\_day(lv\_obj\_t \*obj, uint32\_t day)**

Set the today's year

**Parametri**

- **obj** -- pointer to a calendar object
- **day** -- today's day [1..31]

**void lv\_calendar\_set\_month\_shown(lv\_obj\_t \*obj, uint32\_t year, uint32\_t month)**

Set the currently shown year and month at once

**Parametri**

- **obj** -- pointer to a calendar object
- **year** -- shown year
- **month** -- shown month [1..12]

**void lv\_calendar\_set\_shown\_year(lv\_obj\_t \*obj, uint32\_t year)**

Set the currently shown year

**Parametri**

- **obj** -- pointer to a calendar object
- **year** -- shown year

**void lv\_calendar\_set\_shown\_month(lv\_obj\_t \*obj, uint32\_t month)**

Set the currently shown month

**Parametri**

- **obj** -- pointer to a calendar object
- **month** -- shown month [1..12]

**void lv\_calendar\_set\_highlighted\_dates(lv\_obj\_t \*obj, *lv\_calendar\_date\_t* highlighted[], size\_t date\_num)**

Set the highlighted dates

**Parametri**

- **obj** -- pointer to a calendar object
- **highlighted** -- pointer to an *lv\_calendar\_date\_t* array containing the dates. Only the pointer will be saved so this variable can't be local which will be destroyed later.
- **date\_num** -- number of dates in the array

**void lv\_calendar\_set\_day\_names(lv\_obj\_t \*obj, const char \*\*day\_names)**

Set the name of the days

**Parametri**

- **obj** -- pointer to a calendar object
- **day\_names** -- pointer to an array with the names. E.g. `const char *days[7] = {"Sun", "Mon", ...}` Only the pointer will be saved so this variable can't be local which will be destroyed later.

**lv\_obj\_t \*lv\_calendar\_get\_btnmatrix(const lv\_obj\_t \*obj)**

Get the button matrix object of the calendar. It shows the dates and day names.

**Parametri**

- **obj** -- pointer to a calendar object

**Ritorna**

pointer to a the button matrix

**const *lv\_calendar\_date\_t* \*lv\_calendar\_get\_today\_date(const lv\_obj\_t \*calendar)**

Get the today's date

**Parametri**

- **calendar** -- pointer to a calendar object

**Ritorna**

return pointer to an `lv_calendar_date_t` variable containing the date of today.

```
const lv_calendar_date_t *lv_calendar_get_showed_date(const lv_obj_t *calendar)
```

Get the currently showed

**Parametri**

`calendar` -- pointer to a calendar object

**Ritorna**

pointer to an `lv_calendar_date_t` variable containing the date is being shown.

```
lv_calendar_date_t *lv_calendar_get_highlighted_dates(const lv_obj_t *calendar)
```

Get the highlighted dates

**Parametri**

`calendar` -- pointer to a calendar object

**Ritorna**

pointer to an `lv_calendar_date_t` array containing the dates.

```
size_t lv_calendar_get_highlighted_dates_num(const lv_obj_t *calendar)
```

Get the number of the highlighted dates

**Parametri**

`calendar` -- pointer to a calendar object

**Ritorna**

number of highlighted days

```
lv_result_t lv_calendar_get_pressed_date(const lv_obj_t *calendar, lv_calendar_date_t *date)
```

Get the currently pressed day

**Parametri**

- `calendar` -- pointer to a calendar object
- `date` -- store the pressed date here

**Ritorna**

LV\_RESULT\_OK: there is a valid pressed date  
LV\_RESULT\_INVALID: there is no pressed data

**Variables**

```
const lv_obj_class_t lv_calendar_class
```

```
struct lv_calendar_date_t
```

#include <lv\_calendar.h> Represents a date on the calendar object (platform-agnostic).

**Public Members**

uint16\_t **year**

uint8\_t **month**

1..12

uint8\_t **day**

1..31

**lv\_calendar\_chinese.h****Functions**

```
void lv_calendar_set_chinese_mode(lv_obj_t *obj, bool en)
```

Enable the chinese calendar.

**Parametri**

- **obj** -- pointer to a calendar object.
- **en** -- true: enable chinese calendar; false: disable

```
const char *lv_calendar_get_day_name(lv_calendar_date_t *gregorian)
```

Get the name of the day

**Parametri**

**gregorian** -- to obtain the gregorian time for the name

**Ritorna**

return the name of the day

```
void lv_calendar_gregorian_to_chinese(lv_calendar_date_t *gregorian_time, lv_calendar_chinese_t *chinese_time)
```

Get the chinese time of the gregorian time (reference: <https://www.cnblogs.com/liyang31tg/p/4123171.html>)

**Parametri**

- **gregorian\_time** -- need to convert to chinese time in gregorian time
- **chinese\_time** -- the chinese time convert from gregorian time

```
struct lv_calendar_chinese_t
```

**Public Members**

*lv\_calendar\_date\_t* **today**

bool **leap\_month**

**lv\_calendar\_header\_arrow.h**

**Functions**

```
lv_obj_t *lv_calendar_add_header_arrow(lv_obj_t *parent)
```

Create a calendar header with drop-downs to select the year and month

**Parametri**

**parent** -- pointer to a calendar object.

**Ritorna**

the created header

**Variables**

```
const lv_obj_class_t lv_calendar_header_arrow_class
```

**lv\_calendar\_header\_dropdown.h**

**Functions**

```
lv_obj_t *lv_calendar_add_header_dropdown(lv_obj_t *parent)
```

Create a calendar header with drop-downs to select the year and month

**Parametri**

**parent** -- pointer to a calendar object.

**Ritorna**

the created header

```
void lv_calendar_header_dropdown_set_year_list(lv_obj_t *parent, const char *years_list)
```

Sets a custom calendar year list

**Parametri**

- **parent** -- pointer to a calendar object

- **years\_list** -- pointer to an const char array with the years list, see lv\_dropdown set\_options for more information. E.g. `const char \* years = "2023\n2022\n2021\n2020\n2019" Only the pointer will be saved so this variable can't be local which will be destroyed later.

## Variables

const lv\_obj\_class\_t **lv\_calendar\_header\_dropdown\_class**

**lv\_calendar\_private.h**

```
struct _lv_calendar_t
#include <lv_calendar_private.h> Data of calendar
```

### Public Members

lv\_obj\_t **obj**

lv\_obj\_t \***btm**

*lv\_calendar\_date\_t* **today**

Date of today

*lv\_calendar\_date\_t* **showed\_date**

Currently visible month (day is ignored)

*lv\_calendar\_date\_t* \***highlighted\_dates**

Apply different style on these days (pointer to user-defined array)

size\_t **highlighted\_dates\_num**

Number of elements in **highlighted\_dates**

const char \***map**[8 \* 7]

bool **use\_chinese\_calendar**

char **nums**[7 \* 6][20]

7 \* 6: A week has 7 days, and the calendar displays 6 weeks in total. 20: Including the number of dates, line breaks, names for each day, and reserving several spaces for addresses.

## 14.24.9 canvas

**lv\_canvas.h**

### Defines

**LV\_CANVAS\_BUF\_SIZE**(w, h, bpp, stride)

## Functions

`lv_obj_t *lv_canvas_create(lv_obj_t *parent)`

Create a canvas object

### Parametri

• **parent** -- pointer to an object, it will be the parent of the new canvas

### Ritorna

pointer to the created canvas

`void lv_canvas_set_buffer(lv_obj_t *obj, void *buf, int32_t w, int32_t h, lv_color_format_t cf)`

Set a buffer for the canvas.

Use `lv_canvas_set_draw_buf()` instead if you need to set a buffer with alignment requirement.

### Parametri

- **obj** -- pointer to a canvas object
- **buf** -- buffer where content of canvas will be. The required size is `(lv_image_color_format_get_px_size(cf) * w) / 8 * h` It can be allocated with `lv_malloc()` or it can be statically allocated array (e.g. static `lv_color_t buf[100*50]`) or it can be an address in RAM or external SRAM
- **w** -- width of canvas
- **h** -- height of canvas
- **cf** -- color format. `LV_COLOR_FORMAT...`

`void lv_canvas_set_draw_buf(lv_obj_t *obj, lv_draw_buf_t *draw_buf)`

Set a draw buffer for the canvas. A draw buffer either can be allocated by `lv_draw_buf_create()` or defined statically by `LV_DRAW_BUF_DEFINE_STATIC`. When buffer start address and stride has alignment requirement, it's recommended to use `lv_draw_buf_create`.

### Parametri

- **obj** -- pointer to a canvas object
- **draw\_buf** -- pointer to a draw buffer

`void lv_canvas_set_px(lv_obj_t *obj, int32_t x, int32_t y, lv_color_t color, lv_opa_t opa)`

Set a pixel's color and opacity

### Nota

The following color formats are supported `LV_COLOR_FORMAT_I1/2/4/8`, `LV_COLOR_FORMAT_A8`, `LV_COLOR_FORMAT_RGB565`, `LV_COLOR_FORMAT_RGB888`, `LV_COLOR_FORMAT_XRGB8888`, `LV_COLOR_FORMAT_ARGB8888`

### Parametri

- **obj** -- pointer to a canvas
- **x** -- X coordinate of the pixel
- **y** -- Y coordinate of the pixel
- **color** -- the color
- **opa** -- the opacity

`void lv_canvas_set_palette(lv_obj_t *obj, uint8_t index, lv_color32_t color)`

Set the palette color of a canvas for index format. Valid only for `LV_COLOR_FORMAT_I1/2/4/8`

### Parametri

- **obj** -- pointer to canvas object
- **index** -- the palette color to set:
  - for `LV_COLOR_FORMAT_I1`: 0..1
  - for `LV_COLOR_FORMAT_I2`: 0..3
  - for `LV_COLOR_FORMAT_I4`: 0..15
  - for `LV_COLOR_FORMAT_I8`: 0..255
- **color** -- the color to set

`lv_draw_buf_t *lv_canvas_get_draw_buf(lv_obj_t *obj)`

`lv_color32_t lv_canvas_get_px(lv_obj_t *obj, int32_t x, int32_t y)`

Get a pixel's color and opacity

**Parametri**

- **obj** -- pointer to a canvas
- **x** -- X coordinate of the pixel
- **y** -- Y coordinate of the pixel

**Ritorna**

ARGB8888 color of the pixel

`lv_image_dsc_t *lv_canvas_get_image(lv_obj_t *canvas)`

Get the image of the canvas as a pointer to an `lv_image_dsc_t` variable.

**Parametri**

- **canvas** -- pointer to a canvas object

**Ritorna**

pointer to the image descriptor.

`const void *lv_canvas_get_buf(lv_obj_t *canvas)`

Return the pointer for the buffer. It's recommended to use this function instead of the buffer form the return value of `lv_canvas_get_image()` as is can be aligned

**Parametri**

- **canvas** -- pointer to a canvas object

**Ritorna**

pointer to the buffer

`void lv_canvas_copy_buf(lv_obj_t *obj, const lv_area_t *canvas_area, lv_draw_buf_t *dest_buf, const lv_area_t *dest_area)`

Copy a buffer to the canvas

**Parametri**

- **obj** -- pointer to a canvas object
- **canvas\_area** -- the area of the canvas to copy
- **dest\_buf** -- pointer to a buffer to store the copied data
- **dest\_area** -- the area of the destination buffer to copy to. If omitted NULL, copy to the whole **dest\_buf**

`void lv_canvas_fill_bg(lv_obj_t *obj, lv_color_t color, lv_opa_t opa)`

Fill the canvas with color

**Parametri**

- **obj** -- pointer to a canvas
- **color** -- the background color
- **opa** -- the desired opacity

`void lv_canvas_init_layer(lv_obj_t *canvas, lv_layer_t *layer)`

Initialize a layer to use LVGL's generic draw functions (lv\_draw\_rect/label/...) on the canvas. Needs to be used in pair with `lv_canvas_finish_layer`.

**Parametri**

- **canvas** -- pointer to a canvas
- **layer** -- pointer to a layer variable to initialize

`void lv_canvas_finish_layer(lv_obj_t *canvas, lv_layer_t *layer)`

Wait until all the drawings are finished on layer. Needs to be used in pair with `lv_canvas_init_layer`.

**Parametri**

- **canvas** -- pointer to a canvas
- **layer** -- pointer to a layer to finalize

`uint32_t lv_canvas_buf_size(int32_t w, int32_t h, uint8_t bpp, uint8_t stride)`

Just a wrapper to `LV_CANVAS_BUF_SIZE` for bindings.

## Variables

const lv\_obj\_class\_t **lv\_canvas\_class**

### **lv\_canvas\_private.h**

```
struct _lv_canvas_t
#include <lv_canvas_private.h> Canvas data
```

#### Public Members

lv\_image\_t **img**

lv\_draw\_buf\_t \***draw\_buf**

lv\_draw\_buf\_t **static\_buf**

## 14.24.10 chart

### **lv\_chart.h**

#### Defines

##### **LV\_CHART\_POINT\_NONE**

Default value of points. Can be used to not draw a point

#### Enums

enum **lv\_chart\_type\_t**

Chart types

*Values:*

enumerator **LV\_CHART\_TYPE\_NONE**

Don't draw the series

enumerator **LV\_CHART\_TYPE\_LINE**

Connect the points with lines

enumerator **LV\_CHART\_TYPE\_BAR**

Draw bars for each series

enumerator **LV\_CHART\_TYPE\_STACKED**

Draw a single stacked bar for each data point. Supports only positive values

enumerator **LV\_CHART\_TYPE\_SCATTER**

Draw points and lines in 2D (x,y coordinates)

enum **lv\_chart\_update\_mode\_t**

Chart update mode for `lv_chart_set_next`

*Values:*

**enumerator LV\_CHART\_UPDATE\_MODE\_SHIFT**

Shift old data to the left and add the new one the right

**enumerator LV\_CHART\_UPDATE\_MODE\_CIRCULAR**

Add the new data in a circular way

**enum lv\_chart\_axis\_t**

Enumeration of the axis'

*Values:*

**enumerator LV\_CHART\_AXIS\_PRIMARY\_Y****enumerator LV\_CHART\_AXIS\_SECONDARY\_Y****enumerator LV\_CHART\_AXIS\_PRIMARY\_X****enumerator LV\_CHART\_AXIS\_SECONDARY\_X****enumerator LV\_CHART\_AXIS\_LAST****Functions**

**lv\_obj\_t \*lv\_chart\_create(lv\_obj\_t \*parent)**

Create a chart object

**Parametri**

- **parent** -- pointer to an object, it will be the parent of the new chart

**Ritorna**

pointer to the created chart

**void lv\_chart\_set\_type(lv\_obj\_t \*obj, lv\_chart\_type\_t type)**

Set a new type for a chart

**Parametri**

- **obj** -- pointer to a chart object
- **type** -- new type of the chart (from '*lv\_chart\_type\_t*' enum)

**void lv\_chart\_set\_point\_count(lv\_obj\_t \*obj, uint32\_t cnt)**

Set the number of points on a data line on a chart

**Parametri**

- **obj** -- pointer to a chart object
- **cnt** -- new number of points on the data lines

**void lv\_chart\_set\_axis\_range(lv\_obj\_t \*obj, lv\_chart\_axis\_t axis, int32\_t min, int32\_t max)**

Set the minimal and maximal y values on an axis

**Parametri**

- **obj** -- pointer to a chart object
- **axis** -- LV\_CHART\_AXIS\_PRIMARY\_Y or LV\_CHART\_AXIS\_SECONDARY\_Y
- **min** -- minimum value of the y axis
- **max** -- maximum value of the y axis

**void lv\_chart\_set\_axis\_min\_value(lv\_obj\_t \*obj, lv\_chart\_axis\_t axis, int32\_t min)**

Set the minimal values on an axis

**Parametri**

- **obj** -- pointer to a chart object

- **axis** -- LV\_CHART\_AXIS\_PRIMARY\_Y or LV\_CHART\_AXIS\_SECONDARY\_Y
- **min** -- minimal value of the y axis

**void lv\_chart\_set\_axis\_max\_value(lv\_obj\_t \*obj, lv\_chart\_axis\_t axis, int32\_t max)**

Set the maximal y values on an axis

**Parametri**

- **obj** -- pointer to a chart object
- **axis** -- LV\_CHART\_AXIS\_PRIMARY\_Y or LV\_CHART\_AXIS\_SECONDARY\_Y
- **max** -- maximum value of the y axis

**void lv\_chart\_set\_update\_mode(lv\_obj\_t \*obj, lv\_chart\_update\_mode\_t update\_mode)**

Set update mode of the chart object. Affects

**Parametri**

- **obj** -- pointer to a chart object
- **update\_mode** -- the update mode

**void lv\_chart\_set\_div\_line\_count(lv\_obj\_t \*obj, uint32\_t hdiv, uint32\_t vdiv)**

Set the number of horizontal and vertical division lines

**Parametri**

- **obj** -- pointer to a chart object
- **hdiv** -- number of horizontal division lines
- **vdiv** -- number of vertical division lines

**void lv\_chart\_set\_hor\_div\_line\_count(lv\_obj\_t \*obj, uint32\_t cnt)**

Set the number of horizontal division lines

**Parametri**

- **obj** -- pointer to a chart object
- **cnt** -- number of horizontal division lines

**void lv\_chart\_set\_ver\_div\_line\_count(lv\_obj\_t \*obj, uint32\_t cnt)**

Set the number of vertical division lines

**Parametri**

- **obj** -- pointer to a chart object
- **cnt** -- number of vertical division lines

**lv\_chart\_type\_t lv\_chart\_get\_type(const lv\_obj\_t \*obj)**

Get the type of a chart

**Parametri**

**obj** -- pointer to chart object

**Ritorna**

type of the chart (from 'lv\_chart\_t' enum)

**uint32\_t lv\_chart\_get\_point\_count(const lv\_obj\_t \*obj)**

Get the data point number per data line on chart

**Parametri**

**obj** -- pointer to chart object

**Ritorna**

point number on each data line

**uint32\_t lv\_chart\_get\_x\_start\_point(const lv\_obj\_t \*obj, lv\_chart\_series\_t \*ser)**

Get the current index of the x-axis start point in the data array

**Parametri**

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'

**Ritorna**

the index of the current x start point in the data array

**void lv\_chart\_get\_point\_pos\_by\_id(lv\_obj\_t \*obj, lv\_chart\_series\_t \*ser, uint32\_t id, lv\_point\_t \*p\_out)**

Get the position of a point to the chart.

**Parametri**

- **obj** -- pointer to a chart object
- **ser** -- pointer to series
- **id** -- the index.
- **p\_out** -- store the result position here

**void lv\_chart\_refresh(lv\_obj\_t \*obj)**

Refresh a chart if its data line has changed

**Parametri**

- **obj** -- pointer to chart object

**lv\_chart\_series\_t \*lv\_chart\_add\_series(lv\_obj\_t \*obj, lv\_color\_t color, lv\_chart\_axis\_t axis)**

Allocate and add a data series to the chart

**Parametri**

- **obj** -- pointer to a chart object
- **color** -- color of the data series
- **axis** -- the y axis to which the series should be attached (::LV\_CHART\_AXIS\_PRIMARY\_Y or ::LV\_CHART\_AXIS\_SECONDARY\_Y)

**Ritorna**

pointer to the allocated data series or NULL on failure

**void lv\_chart\_remove\_series(lv\_obj\_t \*obj, lv\_chart\_series\_t \*series)**

Deallocate and remove a data series from a chart

**Parametri**

- **obj** -- pointer to a chart object
- **series** -- pointer to a data series on 'chart'

**void lv\_chart\_hide\_series(lv\_obj\_t \*chart, lv\_chart\_series\_t \*series, bool hide)**

Hide/Unhide a single series of a chart.

**Parametri**

- **chart** -- pointer to a chart object.
- **series** -- pointer to a series object
- **hide** -- true: hide the series

**void lv\_chart\_set\_series\_color(lv\_obj\_t \*chart, lv\_chart\_series\_t \*series, lv\_color\_t color)**

Change the color of a series

**Parametri**

- **chart** -- pointer to a chart object.
- **series** -- pointer to a series object
- **color** -- the new color of the series

**lv\_color\_t lv\_chart\_get\_series\_color(lv\_obj\_t \*chart, const lv\_chart\_series\_t \*series)**

Get the color of a series

**Parametri**

- **chart** -- pointer to a chart object.
- **series** -- pointer to a series object

**Ritorna**

the color of the series

**void lv\_chart\_set\_x\_start\_point(lv\_obj\_t \*obj, lv\_chart\_series\_t \*ser, uint32\_t id)**

Set the index of the x-axis start point in the data array. This point will be considered the first (left) point and the other points will be drawn after it.

**Parametri**

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **id** -- the index of the x point in the data array

`lv_chart_series_t *lv_chart_get_series_next(const lv_obj_t *chart, const lv_chart_series_t *ser)`

Get the next series.

**Parametri**

- **chart** -- pointer to a chart
- **ser** -- the previous series or NULL to get the first

**Ritorna**

the next series or NULL if there is no more.

`lv_chart_cursor_t *lv_chart_add_cursor(lv_obj_t *obj, lv_color_t color, lv_dir_t dir)`

Add a cursor with a given color

**Parametri**

- **obj** -- pointer to chart object
- **color** -- color of the cursor
- **dir** -- direction of the cursor. LV\_DIR\_RIGHT/LEFT/TOP/DOWN/HOR/VER/ALL. OR-ed values are possible

**Ritorna**

pointer to the created cursor

`void lv_chart_set_cursor_pos(lv_obj_t *chart, lv_chart_cursor_t *cursor, lv_point_t *pos)`

Set the coordinate of the cursor with respect to the paddings

**Parametri**

- **chart** -- pointer to a chart object
- **cursor** -- pointer to the cursor
- **pos** -- the new coordinate of cursor relative to the chart

`void lv_chart_set_cursor_pos_x(lv_obj_t *chart, lv_chart_cursor_t *cursor, int32_t x)`

Set the X coordinate of the cursor with respect to the paddings

**Parametri**

- **chart** -- pointer to a chart object
- **cursor** -- pointer to the cursor
- **x** -- the new X coordinate of cursor relative to the chart

`void lv_chart_set_cursor_pos_y(lv_obj_t *chart, lv_chart_cursor_t *cursor, int32_t y)`

Set the coordinate of the cursor with respect to the paddings

**Parametri**

- **chart** -- pointer to a chart object
- **cursor** -- pointer to the cursor
- **y** -- the new Y coordinate of cursor relative to the chart

`void lv_chart_set_cursor_point(lv_obj_t *chart, lv_chart_cursor_t *cursor, lv_chart_series_t *ser, uint32_t point_id)`

Stick the cursor to a point

**Parametri**

- **chart** -- pointer to a chart object
- **cursor** -- pointer to the cursor
- **ser** -- pointer to a series
- **point\_id** -- the point's index or LV\_CHART\_POINT\_NONE to not assign to any points.

`lv_point_t lv_chart_get_cursor_point(lv_obj_t *chart, lv_chart_cursor_t *cursor)`

Get the coordinate of the cursor with respect to the paddings

**Parametri**

- **chart** -- pointer to a chart object
- **cursor** -- pointer to cursor

**Ritorna**

coordinate of the cursor as `lv_point_t`

`void lv_chart_set_all_values(lv_obj_t *obj, lv_chart_series_t *ser, int32_t value)`

Initialize all data points of a series with a value

**Parametri**

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **value** -- the new value for all points. `LV_CHART_POINT_NONE` can be used to hide the points.

`void lv_chart_set_next_value(lv_obj_t *obj, lv_chart_series_t *ser, int32_t value)`

Set the next point's Y value according to the update mode policy.

**Parametri**

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **value** -- the new value of the next data

`void lv_chart_set_next_value2(lv_obj_t *obj, lv_chart_series_t *ser, int32_t x_value, int32_t y_value)`

Set the next point's X and Y value according to the update mode policy.

**Parametri**

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **x\_value** -- the new X value of the next data
- **y\_value** -- the new Y value of the next data

`void lv_chart_set_series_values(lv_obj_t *obj, lv_chart_series_t *ser, const int32_t values[], size_t values_cnt)`

Same as `lv_chart_set_next_value` but set the values from an array

**Parametri**

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **values** -- the new values to set
- **values\_cnt** -- number of items in **values**

`void lv_chart_set_series_values2(lv_obj_t *obj, lv_chart_series_t *ser, const int32_t x_values[], const int32_t y_values[], size_t values_cnt)`

Same as `lv_chart_set_next_value2` but set the values from an array

**Parametri**

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **x\_values** -- the new values to set on the X axis
- **y\_values** -- the new values to set on the Y axis
- **values\_cnt** -- number of items in **x\_values** and **y\_values**

`void lv_chart_set_series_value_by_id(lv_obj_t *obj, lv_chart_series_t *ser, uint32_t id, int32_t value)`

Set an individual point's y value of a chart's series directly based on its index

**Parametri**

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **id** -- the index of the x point in the array
- **value** -- value to assign to array point

`void lv_chart_set_series_value_by_id2(lv_obj_t *obj, lv_chart_series_t *ser, uint32_t id, int32_t x_value, int32_t y_value)`

Set an individual point's x and y value of a chart's series directly based on its index Can be used only with `LV_CHART_TYPE_SCATTER`.

**Parametri**

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **id** -- the index of the x point in the array
- **x\_value** -- the new X value of the next data
- **y\_value** -- the new Y value of the next data

**void lv\_chart\_set\_series\_ext\_y\_array(lv\_obj\_t \*obj, lv\_chart\_series\_t \*ser, int32\_t array[])**

Set an external array for the y data points to use for the chart NOTE: It is the users responsibility to make sure the **point\_cnt** matches the external array size.

**Parametri**

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **array** -- external array of points for chart

**void lv\_chart\_set\_series\_ext\_x\_array(lv\_obj\_t \*obj, lv\_chart\_series\_t \*ser, int32\_t array[])**

Set an external array for the x data points to use for the chart NOTE: It is the users responsibility to make sure the **point\_cnt** matches the external array size.

**Parametri**

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **array** -- external array of points for chart

**int32\_t \*lv\_chart\_get\_series\_y\_array(const lv\_obj\_t \*obj, lv\_chart\_series\_t \*ser)**

Get the array of y values of a series

**Parametri**

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'

**Ritorna**

the array of values with 'point\_count' elements

**int32\_t \*lv\_chart\_get\_series\_x\_array(const lv\_obj\_t \*obj, lv\_chart\_series\_t \*ser)**

Get the array of x values of a series

**Parametri**

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'

**Ritorna**

the array of values with 'point\_count' elements

**uint32\_t lv\_chart\_get\_pressed\_point(const lv\_obj\_t \*obj)**

Get the index of the currently pressed point. It's the same for every series.

**Parametri**

**obj** -- pointer to a chart object

**Ritorna**

the index of the point [0 .. point count] or LV\_CHART\_POINT\_ID\_NONE if no point is being pressed

**int32\_t lv\_chart\_get\_first\_point\_center\_offset(lv\_obj\_t \*obj)**

Get the overall offset from the chart's side to the center of the first point. In case of a bar chart it will be the center of the first column group

**Parametri**

**obj** -- pointer to a chart object

**Ritorna**

the offset of the center

## Variables

const lv\_obj\_class\_t **lv\_chart\_class**

## lv\_chart\_private.h

struct **\_lv\_chart\_series\_t**

#include <lv\_chart\_private.h> Descriptor a chart series

**Public Members**

```
int32_t *x_points  
int32_t *y_points  
lv_color_t color  
uint32_t start_point  
uint32_t hidden  
uint32_t x_ext_buf_assigned  
uint32_t y_ext_buf_assigned  
uint32_t x_axis_sec  
uint32_t y_axis_sec  
struct _lv_chart_cursor_t
```

**Public Members**

```
lv_point_t pos  
int32_t point_id  
lv_color_t color  
lv_chart_series_t *ser  
lv_dir_t dir  
uint32_t pos_set  
    1: pos is set; 0: point_id is set
```

```
struct _lv_chart_t
```

**Public Members**

```
lv_obj_t obj  
lv_ll_t series_ll  
    Linked list for series (stores lv_chart_series_t)
```

***lv\_ll\_t cursor\_ll***

Linked list for cursors (stores lv\_chart\_cursor\_t)

int32\_t **ymin[2]**

int32\_t **ymax[2]**

int32\_t **xmin[2]**

int32\_t **xmax[2]**

int32\_t **pressed\_point\_id**

uint32\_t **hdiv\_cnt**

Number of horizontal division lines

uint32\_t **vdiv\_cnt**

Number of vertical division lines

uint32\_t **point\_cnt**

Number of points in all series

*lv\_chart\_type\_t type*

Chart type

*lv\_chart\_update\_mode\_t update\_mode*

### 14.24.11 checkbox

#### lv\_checkbox.h

##### Functions

lv\_obj\_t \***lv\_checkbox\_create**(lv\_obj\_t \*parent)

Create a check box object

##### Parametri

- **parent** -- pointer to an object, it will be the parent of the new button

##### Ritorna

- pointer to the created check box

void **lv\_checkbox\_set\_text**(lv\_obj\_t \*obj, const char \*txt)

Set the text of a check box. **txt** will be copied and may be deallocated after this function returns.

##### Parametri

- **obj** -- pointer to a check box
- **txt** -- the text of the check box. NULL to refresh with the current text.

void **lv\_checkbox\_set\_text\_static**(lv\_obj\_t \*obj, const char \*txt)

Set the text of a check box. **txt** must not be deallocated during the life of this checkbox.

##### Parametri

- **obj** -- pointer to a check box
- **txt** -- the text of the check box.

```
const char *lv_checkbox_get_text(const lv_obj_t *obj)
```

Get the text of a check box

**Parametri**

**obj** -- pointer to check box object

**Ritorna**

pointer to the text of the check box

## Variables

```
const lv_obj_class_t lv_checkbox_class
```

**lv\_checkbox\_private.h**

```
struct _lv_checkbox_t
```

### Public Members

lv\_obj\_t **obj**

char \***txt**

uint32\_t **static\_txt**

## 14.24.12 dropdown

**lv\_dropdown.h**

### Defines

**LV\_DROPDOWN\_POS\_LAST**

### Enums

enum **\_lv\_property\_dropdown\_id\_t**

*Values:*

enumerator **LV\_PROPERTY\_DROPDOWN\_TEXT**

enumerator **LV\_PROPERTY\_DROPDOWN\_OPTIONS**

enumerator **LV\_PROPERTY\_DROPDOWN\_OPTION\_COUNT**

enumerator **LV\_PROPERTY\_DROPDOWN\_SELECTED**

enumerator **LV\_PROPERTY\_DROPDOWN\_DIR**

enumerator **LV\_PROPERTY\_DROPDOWN\_SYMBOL**

enumerator **LV\_PROPERTY\_DROPDOWN\_SELECTED\_HIGHLIGHT**

enumerator **LV\_PROPERTY\_DROPDOWN\_LIST**

enumerator **LV\_PROPERTY\_DROPDOWN\_IS\_OPEN**

enumerator **LV\_PROPERTY\_DROPDOWN\_END**

## Functions

`lv_obj_t *lv_dropdown_create(lv_obj_t *parent)`

Create a drop-down list object

### Parametri

- **parent** -- pointer to an object, it will be the parent of the new drop-down list

### Ritorna

pointer to the created drop-down list

`void lv_dropdown_set_text(lv_obj_t *obj, const char *txt)`

Set text of the drop-down list's button. If set to `NULL` the selected option's text will be displayed on the button.  
If set to a specific text then that text will be shown regardless of the selected option.

### Parametri

- **obj** -- pointer to a drop-down list object
- **txt** -- the text as a string (Only its pointer is saved)

`void lv_dropdown_set_options(lv_obj_t *obj, const char *options)`

Set the options in a drop-down list from a string. The options will be copied and saved in the object so the `options` can be destroyed after calling this function

### Parametri

- **obj** -- pointer to drop-down list object
- **options** -- a string with '  
' separated options. E.g. "One\nTwo\nThree"

`void lv_dropdown_set_options_static(lv_obj_t *obj, const char *options)`

Set the options in a drop-down list from a static string (global, static or dynamically allocated). Only the pointer of the option string will be saved.

### Parametri

- **obj** -- pointer to drop-down list object
- **options** -- a static string with '  
' separated options. E.g. "One\nTwo\nThree"

`void lv_dropdown_add_option(lv_obj_t *obj, const char *option, uint32_t pos)`

Add an options to a drop-down list from a string. Only works for non-static options.

### Parametri

- **obj** -- pointer to drop-down list object
- **option** -- a string without '  
' E.g. "Four"
- **pos** -- the insert position, indexed from 0, `LV_DROPDOWN_POS_LAST` = end of string

`void lv_dropdown_clear_options(lv_obj_t *obj)`

Clear all options in a drop-down list. Works with both static and dynamic options.

### Parametri

**obj** -- pointer to drop-down list object

`void lv_dropdown_set_selected(lv_obj_t *obj, uint32_t sel_opt)`

Set the selected option

### Parametri

- **obj** -- pointer to drop-down list object

- **sel\_opt** -- id of the selected option (0 ... number of option - 1);

**void lv\_dropdown\_set\_dir(lv\_obj\_t \*obj, lv\_dir\_t dir)**

Set the direction of the a drop-down list

**Parametri**

- **obj** -- pointer to a drop-down list object
- **dir** -- LV\_DIR\_LEFT/RIGHT/TOP/BOTTOM

**void lv\_dropdown\_set\_symbol(lv\_obj\_t \*obj, const void \*symbol)**

Set an arrow or other symbol to display when on drop-down list's button. Typically a down caret or arrow.

**Nota**

angle and zoom transformation can be applied if the symbol is an image. E.g. when drop down is checked (opened) rotate the symbol by 180 degree

**Parametri**

- **obj** -- pointer to drop-down list object
- **symbol** -- a text like LV\_SYMBOL\_DOWN, an image (pointer or path) or NULL to not draw symbol icon

**void lv\_dropdown\_set\_selected\_highlight(lv\_obj\_t \*obj, bool en)**

Set whether the selected option in the list should be highlighted or not

**Parametri**

- **obj** -- pointer to drop-down list object
- **en** -- true: highlight enabled; false: disabled

**lv\_obj\_t \*lv\_dropdown\_get\_list(lv\_obj\_t \*obj)**

Get the list of a drop-down to allow styling or other modifications

**Parametri**

**obj** -- pointer to a drop-down list object

**Ritorna**

pointer to the list of the drop-down

**const char \*lv\_dropdown\_get\_text(lv\_obj\_t \*obj)**

Get text of the drop-down list's button.

**Parametri**

**obj** -- pointer to a drop-down list object

**Ritorna**

the text as string, NULL if no text

**const char \*lv\_dropdown\_get\_options(const lv\_obj\_t \*obj)**

Get the options of a drop-down list

**Parametri**

**obj** -- pointer to drop-down list object

**Ritorna**

the options separated by '

'-s (E.g. "Option1\nOption2\nOption3")

**uint32\_t lv\_dropdown\_get\_selected(const lv\_obj\_t \*obj)**

Get the index of the selected option

**Parametri**

**obj** -- pointer to drop-down list object

**Ritorna**

index of the selected option (0 ... number of option - 1);

**uint32\_t lv\_dropdown\_get\_option\_count(const lv\_obj\_t \*obj)**

Get the total number of options

**Parametri**  
**obj** -- pointer to drop-down list object  
**Ritorna**  
the total number of options in the list

void **lv\_dropdown\_get\_selected\_str**(const lv\_obj\_t \*obj, char \*buf, uint32\_t buf\_size)

Get the current selected option as a string

**Parametri**

- **obj** -- pointer to drop-down object
- **buf** -- pointer to an array to store the string
- **buf\_size** -- size of **buf** in bytes. 0: to ignore it.

int32\_t **lv\_dropdown\_get\_option\_index**(lv\_obj\_t \*obj, const char \*option)

Get the index of an option.

**Parametri**

- **obj** -- pointer to drop-down object
- **option** -- an option as string

**Ritorna**

index of **option** in the list of all options. -1 if not found.

const char \***lv\_dropdown\_get\_symbol**(lv\_obj\_t \*obj)

Get the symbol on the drop-down list. Typically a down caret or arrow.

**Parametri**

- **obj** -- pointer to drop-down list object

**Ritorna**

the symbol or NULL if not enabled

bool **lv\_dropdown\_get\_selected\_highlight**(lv\_obj\_t \*obj)

Get whether the selected option in the list should be highlighted or not

**Parametri**

- **obj** -- pointer to drop-down list object

**Ritorna**

true: highlight enabled; false: disabled

*lv\_dir\_t* **lv\_dropdown\_get\_dir**(const lv\_obj\_t \*obj)

Get the direction of the drop-down list

**Parametri**

- **obj** -- pointer to a drop-down list object

**Ritorna**

LV\_DIR\_LEFT/RIGHT/TOP/BOTTOM

void **lv\_dropdown\_open**(lv\_obj\_t \*dropdown\_obj)

Open the drop-down list

**Parametri**

- **dropdown\_obj** -- pointer to drop-down list object

void **lv\_dropdown\_close**(lv\_obj\_t \*obj)

Close (Collapse) the drop-down list

**Parametri**

- **obj** -- pointer to drop-down list object

bool **lv\_dropdown\_is\_open**(lv\_obj\_t \*obj)

Tells whether the list is opened or not

**Parametri**

- **obj** -- pointer to a drop-down list object

**Ritorna**

true if the list is opened

## Variables

```
const lv_obj_class_t lv_dropdown_class  
const lv_obj_class_t lv_dropdownlist_class  
lv_dropdown_private.h  
struct _lv_dropdown_t
```

### Public Members

**lv\_obj\_t obj**  
**lv\_obj\_t \*list**  
     The dropped down list

**const char \*text**  
     Text to display on the dropdown's button

**const void \*symbol**  
     Arrow or other icon when the drop-down list is closed

**char \*options**  
     Options in a '  
     ' separated list

**uint32\_t option\_cnt**  
     Number of options

**uint32\_t sel\_opt\_id**  
     Index of the currently selected option

**uint32\_t sel\_opt\_id\_orig**  
     Store the original index on focus

**uint32\_t pr\_opt\_id**  
     Index of the currently pressed option

**uint8\_t dir**  
     Direction in which the list should open

**uint8\_t static\_txt**  
     1: Only a pointer is saved in **options**

**uint8\_t selected\_highlight**  
     1: Make the selected option highlighted in the list

struct **\_lv\_dropdown\_list\_t**

## Public Members

lv\_obj\_t **obj**

lv\_obj\_t \***dropdown**

### 14.24.13 image

[lv\\_image.h](#)

#### Defines

**LV\_IMAGE\_DECLARE**(var\_name)

Use this macro to declare an image in a C file

#### Enums

enum **lv\_image\_align\_t**

Image size mode, when image size and object size is different

*Values:*

enumerator **LV\_IMAGE\_ALIGN\_DEFAULT**

enumerator **LV\_IMAGE\_ALIGN\_TOP\_LEFT**

enumerator **LV\_IMAGE\_ALIGN\_TOP\_MID**

enumerator **LV\_IMAGE\_ALIGN\_TOP\_RIGHT**

enumerator **LV\_IMAGE\_ALIGN\_BOTTOM\_LEFT**

enumerator **LV\_IMAGE\_ALIGN\_BOTTOM\_MID**

enumerator **LV\_IMAGE\_ALIGN\_BOTTOM\_RIGHT**

enumerator **LV\_IMAGE\_ALIGN\_LEFT\_MID**

enumerator **LV\_IMAGE\_ALIGN\_RIGHT\_MID**

enumerator **LV\_IMAGE\_ALIGN\_CENTER**

enumerator **LV\_IMAGE\_ALIGN\_AUTO\_TRANSFORM**

enumerator **LV\_IMAGE\_ALIGN\_STRETCH**

enumerator **LV\_IMAGE\_ALIGN\_TILE**

enumerator **LV\_IMAGE\_ALIGN\_CONTAIN**

enumerator **LV\_IMAGE\_ALIGN\_COVER**

enum **\_lv\_property\_image\_id\_t**

*Values:*

enumerator **LV\_PROPERTY\_IMAGE\_SRC**

enumerator **LV\_PROPERTY\_IMAGE\_OFFSET\_X**

enumerator **LV\_PROPERTY\_IMAGE\_OFFSET\_Y**

enumerator **LV\_PROPERTY\_IMAGE\_ROTATION**

enumerator **LV\_PROPERTY\_IMAGE\_PIVOT**

enumerator **LV\_PROPERTY\_IMAGE\_SCALE**

enumerator **LV\_PROPERTY\_IMAGE\_SCALE\_X**

enumerator **LV\_PROPERTY\_IMAGE\_SCALE\_Y**

enumerator **LV\_PROPERTY\_IMAGE\_BLEND\_MODE**

enumerator **LV\_PROPERTY\_IMAGE\_ANTIALIAS**

enumerator **LV\_PROPERTY\_IMAGE\_INNER\_ALIGN**

enumerator **LV\_PROPERTY\_IMAGE\_END**

## Functions

`lv_obj_t *lv_image_create(lv_obj_t *parent)`

Create an image object

**Parametri**

**parent** -- pointer to an object, it will be the parent of the new image

**Ritorna**

pointer to the created image

`void lv_image_set_src(lv_obj_t *obj, const void *src)`

Set the image data to display on the object

**Parametri**

- **obj** -- pointer to an image object
- **src** -- 1) pointer to an `lv_image_desc_t` descriptor (converted by LVGL's image converter) (e.g. `&my_img`) or 2) path to an image file (e.g. "S:/dir/img.bin") or 3) a SYMBOL (e.g. `LV_SYMBOL_OK`)

`void lv_image_set_offset_x(lv_obj_t *obj, int32_t x)`

Set an offset for the source of an image so the image will be displayed from the new origin.

**Parametri**

- **obj** -- pointer to an image
- **x** -- the new offset along x axis.

**void lv\_image\_set\_offset\_y(lv\_obj\_t \*obj, int32\_t y)**

Set an offset for the source of an image. so the image will be displayed from the new origin.

**Parametri**

- **obj** -- pointer to an image
- **y** -- the new offset along y axis.

**void lv\_image\_set\_rotation(lv\_obj\_t \*obj, int32\_t angle)**

Set the rotation angle of the image. The image will be rotated around the set pivot set by [lv\\_image\\_set\\_pivot\(\)](#) Note that indexed and alpha only images can't be transformed.

 **Nota**

if image\_align is LV\_IMAGE\_ALIGN\_STRETCH or LV\_IMAGE\_ALIGN\_FIT rotation will be set to 0 automatically.

**Parametri**

- **obj** -- pointer to an image object
- **angle** -- rotation in degree with 0.1 degree resolution (0..3600: clock wise)

**void lv\_image\_set\_pivot(lv\_obj\_t \*obj, int32\_t x, int32\_t y)**

Set the rotation center of the image. The image will be rotated around this point. x, y can be set with value of LV\_PCT, lv\_image\_get\_pivot will return the true pixel coordinate of pivot in this case.

**Parametri**

- **obj** -- pointer to an image object
- **x** -- rotation center x of the image
- **y** -- rotation center y of the image

**void lv\_image\_set\_pivot\_x(lv\_obj\_t \*obj, int32\_t x)**

Set the rotation horizontal center of the image.

**Parametri**

- **obj** -- pointer to an image object
- **x** -- rotation center x of the image, or [lv\\_pct\(\)](#)

**void lv\_image\_set\_pivot\_y(lv\_obj\_t \*obj, int32\_t y)**

Set the rotation vertical center of the image.

**Parametri**

- **obj** -- pointer to an image object
- **y** -- rotation center y of the image, or [lv\\_pct\(\)](#)

**void lv\_image\_set\_scale(lv\_obj\_t \*obj, uint32\_t zoom)**

Set the zoom factor of the image. Note that indexed and alpha only images can't be transformed.

**Parametri**

- **obj** -- pointer to an image object
- **zoom** -- the zoom factor. Example values:
  - 256 or LV\_SCALE\_NONE: no zoom
  - <256: scale down
  - >256: scale up
  - 128: half size
  - 512: double size

**void lv\_image\_set\_scale\_x(lv\_obj\_t \*obj, uint32\_t zoom)**

Set the horizontal zoom factor of the image. Note that indexed and alpha only images can't be transformed.

**Parametri**

- **obj** -- pointer to an image object
- **zoom** -- the zoom factor. Example values:
  - 256 or LV\_SCALE\_NONE: no zoom

- <256: scale down
- >256: scale up
- 128: half size
- 512: double size

**void lv\_image\_set\_scale\_y(lv\_obj\_t \*obj, uint32\_t zoom)**

Set the vertical zoom factor of the image. Note that indexed and alpha only images can't be transformed.

**Parametri**

- **obj** -- pointer to an image object
- **zoom** -- the zoom factor. Example values:
  - 256 or LV\_SCALE\_NONE: no zoom
  - <256: scale down
  - >256: scale up
  - 128: half size
  - 512: double size

**void lv\_image\_set\_blend\_mode(lv\_obj\_t \*obj, lv\_blend\_mode\_t blend\_mode)**

Set the blend mode of an image.

**Parametri**

- **obj** -- pointer to an image object
- **blend\_mode** -- the new blend mode

**void lv\_image\_set\_antialias(lv\_obj\_t \*obj, bool antialias)**

Enable/disable anti-aliasing for the transformations (rotate, zoom) or not. The quality is better with anti-aliasing looks better but slower.

**Parametri**

- **obj** -- pointer to an image object
- **antialias** -- true: anti-aliased; false: not anti-aliased

**void lv\_image\_set\_inner\_align(lv\_obj\_t \*obj, lv\_image\_align\_t align)**

Set the image object size mode.

 **Nota**

if image\_align is LV\_IMAGE\_ALIGN\_STRETCH or LV\_IMAGE\_ALIGN\_FIT rotation, scale and pivot will be overwritten and controlled internally.

**Parametri**

- **obj** -- pointer to an image object
- **align** -- the new align mode.

**void lv\_image\_set\_bitmap\_map\_src(lv\_obj\_t \*obj, const lv\_image\_dsc\_t \*src)**

Set an A8 bitmap mask for the image.

**Parametri**

- **obj** -- pointer to an image object
- **src** -- an *lv\_image\_dsc\_t* bitmap mask source.

**const void \*lv\_image\_get\_src(lv\_obj\_t \*obj)**

Get the source of the image

**Parametri**

- **obj** -- pointer to an image object

**Ritorna**

the image source (symbol, file name or ::lv-img\_dsc\_t for C arrays)

`int32_t lv_image_get_offset_x(lv_obj_t *obj)`

Get the offset's x attribute of the image object.

**Parametri**

**obj** -- pointer to an image

**Ritorna**

offset X value.

`int32_t lv_image_get_offset_y(lv_obj_t *obj)`

Get the offset's y attribute of the image object.

**Parametri**

**obj** -- pointer to an image

**Ritorna**

offset Y value.

`int32_t lv_image_get_rotation(lv_obj_t *obj)`

Get the rotation of the image.

**Nota**

if image\_align is LV\_IMAGE\_ALIGN\_STRETCH or LV\_IMAGE\_ALIGN\_FIT rotation will be set to 0 automatically.

**Parametri**

**obj** -- pointer to an image object

**Ritorna**

rotation in 0.1 degrees (0..3600)

`void lv_image_get_pivot(lv_obj_t *obj, lv_point_t *pivot)`

Get the pivot (rotation center) of the image. If pivot is set with LV\_PCT, convert it to px before return.

**Parametri**

- **obj** -- pointer to an image object
- **pivot** -- store the rotation center here

`int32_t lv_image_get_scale(lv_obj_t *obj)`

Get the zoom factor of the image.

**Parametri**

**obj** -- pointer to an image object

**Ritorna**

zoom factor (256: no zoom)

`int32_t lv_image_get_scale_x(lv_obj_t *obj)`

Get the horizontal zoom factor of the image.

**Parametri**

**obj** -- pointer to an image object

**Ritorna**

zoom factor (256: no zoom)

`int32_t lv_image_get_scale_y(lv_obj_t *obj)`

Get the vertical zoom factor of the image.

**Parametri**

**obj** -- pointer to an image object

**Ritorna**

zoom factor (256: no zoom)

`int32_t lv_image_get_src_width(lv_obj_t *obj)`

Get the width of an image before any transformations.

**Parametri**

**obj** -- Pointer to an image object.

**Ritorna**

The width of the image.

`int32_t lv_image_get_src_height(lv_obj_t *obj)`

Get the height of an image before any transformations.

**Parametri**

**obj** -- Pointer to an image object.

**Ritorna**

The height of the image.

`int32_t lv_image_get_transformed_width(lv_obj_t *obj)`

Get the transformed width of an image object.

**Parametri**

**obj** -- Pointer to an image object.

**Ritorna**

The transformed width of the image.

`int32_t lv_image_get_transformed_height(lv_obj_t *obj)`

Get the transformed height of an image object.

**Parametri**

**obj** -- Pointer to an image object.

**Ritorna**

The transformed height of the image.

`lv_blend_mode_t lv_image_get_blend_mode(lv_obj_t *obj)`

Get the current blend mode of the image

**Parametri**

**obj** -- pointer to an image object

**Ritorna**

the current blend mode

`bool lv_image_get_antialias(lv_obj_t *obj)`

Get whether the transformations (rotate, zoom) are anti-aliased or not

**Parametri**

**obj** -- pointer to an image object

**Ritorna**

true: anti-aliased; false: not anti-aliased

`lv_image_align_t lv_image_get_inner_align(lv_obj_t *obj)`

Get the size mode of the image

**Parametri**

**obj** -- pointer to an image object

**Ritorna**

element of `lv_image_align_t`

`const lv_image_dsc_t *lv_image_get_bitmap_map_src(lv_obj_t *obj)`

Get the bitmap mask source.

**Parametri**

**obj** -- pointer to an image object

**Ritorna**

an `lv_image_dsc_t` bitmap mask source.

**Variables**

`const lv_obj_class_t lv_image_class`

**lv\_image\_private.h****struct \_lv\_image\_t**

#include &lt;lv\_image\_private.h&gt; Data of image

**Public Members****lv\_obj\_t obj****const void \*src**

Image source: Pointer to an array or a file or a symbol

**const lv\_image\_dsc\_t \*bitmap\_mask\_src**

Pointer to an A8 bitmap mask

**lv\_point\_t offset****int32\_t w**

Width of the image (Handled by the library)

**int32\_t h**

Height of the image (Handled by the library)

**uint32\_t rotation**

Rotation angle of the image

**uint32\_t scale\_x**

256 means no zoom, 512 double size, 128 half size

**uint32\_t scale\_y**

256 means no zoom, 512 double size, 128 half size

**lv\_point\_t pivot**

Rotation center of the image

**uint32\_t src\_type**See: [lv\\_image\\_src\\_t](#)**uint32\_t cf**Color format from [lv\\_color\\_format\\_t](#)**uint32\_t antialias**

Apply anti-aliasing in transformations (rotate, zoom)

**uint32\_t align**Image size mode when image size and object size is different. See [lv\\_image\\_align\\_t](#)**uint32\_t blend\_mode**Element of [lv\\_blend\\_mode\\_t](#)

## 14.24.14 imagebutton

`lv_imagebutton.h`

### Enums

enum `lv_imagebutton_state_t`

*Values:*

enumerator `LV_IMAGEBUTTON_STATE_RELEASED`

enumerator `LV_IMAGEBUTTON_STATE_PRESSED`

enumerator `LV_IMAGEBUTTON_STATE_DISABLED`

enumerator `LV_IMAGEBUTTON_STATE_CHECKED_RELEASED`

enumerator `LV_IMAGEBUTTON_STATE_CHECKED_PRESSED`

enumerator `LV_IMAGEBUTTON_STATE_CHECKED_DISABLED`

enumerator `LV_IMAGEBUTTON_STATE_NUM`

### Functions

`lv_obj_t *lv_imagebutton_create(lv_obj_t *parent)`

Create an image button object

#### Parametri

- **parent** -- pointer to an object, it will be the parent of the new image button

#### Ritorna

pointer to the created image button

`void lv_imagebutton_set_src(lv_obj_t *imagebutton, lv_imagebutton_state_t state, const void *src_left, const void *src_mid, const void *src_right)`

Set images for a state of the image button

#### Parametri

- **imagebutton** -- pointer to an image button object
- **state** -- for which state set the new image
- **src\_left** -- pointer to an image source for the left side of the button (a C array or path to a file)
- **src\_mid** -- pointer to an image source for the middle of the button (ideally 1px wide) (a C array or path to a file)
- **src\_right** -- pointer to an image source for the right side of the button (a C array or path to a file)

`void lv_imagebutton_set_state(lv_obj_t *imagebutton, lv_imagebutton_state_t state)`

Use this function instead of `lv_obj_add/remove_state` to set a state manually

#### Parametri

- **imagebutton** -- pointer to an image button object
- **state** -- the new state

`const void *lv_imagebutton_get_src_left(lv_obj_t *imagebutton, lv_imagebutton_state_t state)`

Get the left image in a given state

#### Parametri

- **imagebutton** -- pointer to an image button object

- **state** -- the state where to get the image (from `lv_button_state_t`)`

**Ritorna**

pointer to the left image source (a C array or path to a file)

```
const void *lv_imagebutton_get_src_middle(lv_obj_t *imagebutton, lv_imagebutton_state_t state)
```

Get the middle image in a given state

**Parametri**

- **imagebutton** -- pointer to an image button object
- **state** -- the state where to get the image (from `lv_button_state_t`)`

**Ritorna**

pointer to the middle image source (a C array or path to a file)

```
const void *lv_imagebutton_get_src_right(lv_obj_t *imagebutton, lv_imagebutton_state_t state)
```

Get the right image in a given state

**Parametri**

- **imagebutton** -- pointer to an image button object
- **state** -- the state where to get the image (from `lv_button_state_t`)`

**Ritorna**

pointer to the right image source (a C array or path to a file)

**Variables**

```
const lv_obj_class_t lv_imagebutton_class
```

**lv\_imagebutton\_private.h**

```
struct _lv_imagebutton_src_info_t
```

**Public Members**

```
const void *img_src
```

```
lv_image_header_t header
```

```
struct _lv_imagebutton_t
```

```
#include <lv_imagebutton_private.h> Data of image button
```

**Public Members**

```
lv_obj_t obj
```

```
lv_imagebutton_src_info_t src_mid[LV_IMAGEBUTTON_STATE_NUM]
```

Store center images to each state

```
lv_imagebutton_src_info_t src_left[LV_IMAGEBUTTON_STATE_NUM]
```

Store left side images to each state

```
lv_imagebutton_src_info_t src_right[LV_IMAGEBUTTON_STATE_NUM]
```

Store right side images to each state

## 14.24.15 keyboard

`lv_keyboard.h`

### Defines

`LV_KEYBOARD_CTRL_BUTTON_FLAGS`

### Enums

enum `lv_keyboard_mode_t`

Current keyboard mode.

*Values:*

enumerator `LV_KEYBOARD_MODE_TEXT_LOWER`

enumerator `LV_KEYBOARD_MODE_TEXT_UPPER`

enumerator `LV_KEYBOARD_MODE_SPECIAL`

enumerator `LV_KEYBOARD_MODE_NUMBER`

enumerator `LV_KEYBOARD_MODE_USER_1`

enumerator `LV_KEYBOARD_MODE_USER_2`

enumerator `LV_KEYBOARD_MODE_USER_3`

enumerator `LV_KEYBOARD_MODE_USER_4`

enumerator `LV_KEYBOARD_MODE_TEXT_ARABIC`

enum `_lv_property_keyboard_id_t`

*Values:*

enumerator `LV_PROPERTY_KEYBOARD_TEXTAREA`

enumerator `LV_PROPERTY_KEYBOARD_MODE`

enumerator `LV_PROPERTY_KEYBOARD_POPOVERS`

enumerator `LV_PROPERTY_KEYBOARD_SELECTED_BUTTON`

enumerator `LV_PROPERTY_KEYBOARD_END`

## Functions

`lv_obj_t *lv_keyboard_create(lv_obj_t *parent)`

Create a Keyboard object

**Parametri**

**parent** -- pointer to an object, it will be the parent of the new keyboard

**Ritorna**

    pointer to the created keyboard

`void lv_keyboard_set_textarea(lv_obj_t *kb, lv_obj_t *ta)`

Assign a Text Area to the Keyboard. The pressed characters will be put there.

**Parametri**

- **kb** -- pointer to a Keyboard object
- **ta** -- pointer to a Text Area object to write there

`void lv_keyboard_set_mode(lv_obj_t *kb, lv_keyboard_mode_t mode)`

Set a new a mode (text or number map)

**Parametri**

- **kb** -- pointer to a Keyboard object
- **mode** -- the mode from '*lv\_keyboard\_mode\_t*'

`void lv_keyboard_set_popovers(lv_obj_t *kb, bool en)`

Show the button title in a popover when pressed.

**Parametri**

- **kb** -- pointer to a Keyboard object
- **en** -- whether "popovers" mode is enabled

`void lv_keyboard_set_map(lv_obj_t *kb, lv_keyboard_mode_t mode, const char *const map[], const lv_buttonmatrix_ctrl_t ctrl_map[])`

Set a new map for the keyboard

**Parametri**

- **kb** -- pointer to a Keyboard object
- **mode** -- keyboard map to alter '*lv\_keyboard\_mode\_t*'
- **map** -- pointer to a string array to describe the map. See '*lv\_buttonmatrix\_set\_map()*' for more info.
- **ctrl\_map** -- See '*lv\_buttonmatrix\_set\_ctrl\_map()*' for more info.

`lv_obj_t *lv_keyboard_get_textarea(const lv_obj_t *kb)`

Assign a Text Area to the Keyboard. The pressed characters will be put there.

**Parametri**

**kb** -- pointer to a Keyboard object

**Ritorna**

    pointer to the assigned Text Area object

`lv_keyboard_mode_t lv_keyboard_get_mode(const lv_obj_t *kb)`

Set a new a mode (text or number map)

**Parametri**

**kb** -- pointer to a Keyboard object

**Ritorna**

    the current mode from '*lv\_keyboard\_mode\_t*'

`bool lv_keyboard_get_popovers(const lv_obj_t *obj)`

Tell whether "popovers" mode is enabled or not.

**Parametri**

**obj** -- pointer to a Keyboard object

**Ritorna**

    true: "popovers" mode is enabled; false: disabled

`const char *const *lv_keyboard_get_map_array(const lv_obj_t *kb)`

Get the current map of a keyboard

**Parametri**

**kb** -- pointer to a keyboard object

**Ritorna**

the current map

```
uint32_t lv_keyboard_get_selected_button(const lv_obj_t *obj)
```

Get the index of the lastly "activated" button by the user (pressed, released, focused etc) Useful in the `event_cb` to get the text of the button, check if hidden etc.

**Parametri**

**obj** -- pointer to button matrix object

**Ritorna**

index of the last released button (LV\_BUTTONMATRIX\_BUTTON\_NONE: if unset)

```
const char *lv_keyboard_get_button_text(const lv_obj_t *obj, uint32_t btn_id)
```

Get the button's text

**Parametri**

- **obj** -- pointer to button matrix object

- **btn\_id** -- the index a button not counting new line characters.

**Ritorna**

text of btn\_index` button

```
void lv_keyboard_def_event_cb(lv_event_t *e)
```

Default keyboard event to add characters to the Text area and change the map. If a custom `event_cb` is added to the keyboard this function can be called from it to handle the button clicks

**Parametri**

**e** -- the triggering event

**Variables**

```
const lv_obj_class_t lv_keyboard_class
```

**lv\_keyboard\_private.h**

```
struct _lv_keyboard_t
```

```
#include <lv_keyboard_private.h> Data of keyboard
```

**Public Members**

`lv_buttonmatrix_t btm`

`lv_obj_t *ta`

Pointer to the assigned text area

`lv_keyboard_mode_t mode`

Key map type

`uint8_t popovers`

Show button titles in popovers on press

**14.24.16 label****lv\_label.h****Defines**

**LV\_LABEL\_DOT\_NUM****LV\_LABEL\_POS\_LAST****LV\_LABEL\_TEXT\_SELECTION\_OFF****LV\_LABEL\_DEFAULT\_TEXT**

## Enums

enum **lv\_label\_long\_mode\_t**

Long mode behaviors. Used in 'lv\_label\_ext\_t'

*Values:*enumerator **LV\_LABEL\_LONG\_MODE\_WRAP**

Keep the object width, wrap lines longer than object width and expand the object height

enumerator **LV\_LABEL\_LONG\_MODE\_DOTS**

Keep the size and write dots at the end if the text is too long

enumerator **LV\_LABEL\_LONG\_MODE\_SCROLL**

Keep the size and roll the text back and forth

enumerator **LV\_LABEL\_LONG\_MODE\_SCROLL\_CIRCULAR**

Keep the size and roll the text circularly

enumerator **LV\_LABEL\_LONG\_MODE\_CLIP**

Keep the size and clip the text out of it

enum **\_lv\_property\_label\_id\_t***Values:*enumerator **LV\_PROPERTY\_LABEL\_TEXT**enumerator **LV\_PROPERTY\_LABEL\_LONG\_MODE**enumerator **LV\_PROPERTY\_LABEL\_TEXT\_SELECTION\_START**enumerator **LV\_PROPERTY\_LABEL\_TEXT\_SELECTION\_END**enumerator **LV\_PROPERTY\_LABEL\_END**

## Functions

**lv\_obj\_t \*lv\_label\_create(lv\_obj\_t \*parent)**

Create a label object

**Parametri****parent** -- pointer to an object, it will be the parent of the new label.

**Ritorna**

pointer to the created button

**void lv\_label\_set\_text(lv\_obj\_t \*obj, const char \*text)**

Set a new text for a label. Memory will be allocated to store the text by the label.

**Parametri**

- **obj** -- pointer to a label object
- **text** -- '\0' terminated character string. NULL to refresh with the current text.

**void lv\_label\_set\_text\_fmt(lv\_obj\_t \*obj, const char \*fmt, ...)**

Set a new formatted text for a label. Memory will be allocated to store the text by the label.

Example:

```
lv_label_set_text_fmt(label1, "%d user", user_num);
```

**Parametri**

- **obj** -- pointer to a label object
- **fmt** -- printf-like format

**void lv\_label\_set\_text\_vfmt(lv\_obj\_t \*obj, const char \*fmt, va\_list args)**

Set a new formatted text for a label. Memory will be allocated to store the text by the label.

Example:

```
va_list args;
va_start(args, fmt);
lv_label_set_text_vfmt(label1, fmt, args);
va_end(args);
```

**Parametri**

- **obj** -- pointer to a label object
- **fmt** -- printf-like format
- **args** -- variadic arguments list

**void lv\_label\_set\_text\_static(lv\_obj\_t \*obj, const char \*text)**

Set a static text. It will not be saved by the label so the 'text' variable has to be 'alive' while the label exists.

**Parametri**

- **obj** -- pointer to a label object
- **text** -- pointer to a text. NULL to refresh with the current text.

**void lv\_label\_set\_long\_mode(lv\_obj\_t \*obj, lv\_label\_long\_mode\_t long\_mode)**

Set the behavior of the label with text longer than the object size

**Parametri**

- **obj** -- pointer to a label object
- **long\_mode** -- the new mode from 'lv\_label\_long\_mode' enum. In LV\_LONG\_WRAP/DOT/SCROLL/SCROLL\_CIRC the size of the label should be set AFTER this function

**void lv\_label\_set\_text\_selection\_start(lv\_obj\_t \*obj, uint32\_t index)**

Set where text selection should start

**Parametri**

- **obj** -- pointer to a label object
- **index** -- character index from where selection should start. LV\_LABEL\_TEXT\_SELECTION\_OFF for no selection

**void lv\_label\_set\_text\_selection\_end(lv\_obj\_t \*obj, uint32\_t index)**

Set where text selection should end

**Parametri**

- **obj** -- pointer to a label object
- **index** -- character index where selection should end. LV\_LABEL\_TEXT\_SELECTION\_OFF for no selection

**void lv\_label\_set\_recolor(lv\_obj\_t \*obj, bool en)**

Enable the recoloring by in-line commands

**Parametri**

- **obj** -- pointer to a label object
- **en** -- true: enable recoloring, false: disable Example: "This is a #ff0000 red# word"

**char \*lv\_label\_get\_text(const lv\_obj\_t \*obj)**

Get the text of a label

**Parametri**

- **obj** -- pointer to a label object

**Ritorna**

the text of the label

**lv\_label\_long\_mode\_t lv\_label\_get\_long\_mode(const lv\_obj\_t \*obj)**

Get the long mode of a label

**Parametri**

- **obj** -- pointer to a label object

**Ritorna**

the current long mode

**void lv\_label\_get\_letter\_pos(const lv\_obj\_t \*obj, uint32\_t char\_id, lv\_point\_t \*pos)**

Get the relative x and y coordinates of a letter

**Parametri**

- **obj** -- pointer to a label object
- **char\_id** -- index of the character [0 ... text length - 1]. Expressed in character index, not byte index (different in UTF-8)
- **pos** -- store the result here (E.g. index = 0 gives 0;0 coordinates if the text is aligned to the left)

**uint32\_t lv\_label\_get\_letter\_on(const lv\_obj\_t \*obj, lv\_point\_t \*pos\_in, bool bidi)**

Get the index of letter on a relative point of a label.

**Parametri**

- **obj** -- pointer to label object
- **pos\_in** -- pointer to point with coordinates on a the label
- **bidi** -- whether to use bidi processed

**Ritorna**

The index of the letter on the 'pos\_p' point (E.g. on 0;0 is the 0. letter if aligned to the left)  
Expressed in character index and not byte index (different in UTF-8)

**bool lv\_label\_is\_char\_under\_pos(const lv\_obj\_t \*obj, lv\_point\_t \*pos)**

Check if a character is drawn under a point.

**Parametri**

- **obj** -- pointer to a label object
- **pos** -- Point to check for character under

**Ritorna**

whether a character is drawn under the point

**uint32\_t lv\_label\_get\_text\_selection\_start(const lv\_obj\_t \*obj)**

Get the selection start index.

**Parametri**

- **obj** -- pointer to a label object.

**Ritorna**

selection start index. LV\_LABEL\_TEXT\_SELECTION\_OFF if nothing is selected.

**uint32\_t lv\_label\_get\_text\_selection\_end(const lv\_obj\_t \*obj)**

Get the selection end index.

**Parametri**

- **obj** -- pointer to a label object.

**Ritorna**

selection end index. LV\_LABEL\_TXT\_SEL\_OFF if nothing is selected.

`bool lv_label_get_recolor(const lv_obj_t *obj)`

Get the recoloring attribute.

**Parametri**

- **obj** -- pointer to a label object.

**Ritorna**

true: recoloring is enabled, false: recoloring is disabled

`void lv_label_ins_text(lv_obj_t *obj, uint32_t pos, const char *txt)`

Insert a text to a label. The label text cannot be static.

**Parametri**

- **obj** -- pointer to a label object
- **pos** -- character index to insert. Expressed in character index and not byte index. 0: before first char. LV\_LABEL\_POS\_LAST: after last char.
- **txt** -- pointer to the text to insert

`void lv_label_cut_text(lv_obj_t *obj, uint32_t pos, uint32_t cnt)`

Delete characters from a label. The label text cannot be static.

**Parametri**

- **obj** -- pointer to a label object
- **pos** -- character index from where to cut. Expressed in character index and not byte index. 0: start in front of the first character
- **cnt** -- number of characters to cut

## Variables

const lv\_obj\_class\_t **lv\_label\_class**

**lv\_label\_private.h**

struct **\_lv\_label\_t**

### Public Members

**lv\_obj\_t obj**

**char \*text**

**char dot[3 + 1]**

Bytes that have been replaced with dots

**uint32\_t dot\_begin**

Offset where bytes have been replaced with dots

**lv\_draw\_label\_hint\_t hint**

**uint32\_t sel\_start**

**uint32\_t sel\_end**

**lv\_point\_t size\_cache**

Text size cache

***lv\_point\_t* offset**

Text draw position offset

***lv\_label\_long\_mode\_t* long\_mode**

Determine what to do with the long texts

**uint8\_t static\_txt**

Flag to indicate the text is static

**uint8\_t recolor**

Enable in-line letter re-coloring

**uint8\_t expand**

Ignore real width (used by the library with LV\_LABEL\_LONG\_MODE\_SCROLL)

**uint8\_t invalid\_size\_cache**

1: Recalculate size and update cache

***lv\_point\_t* text\_size**

## 14.24.17 led

**lv\_led.h****Defines****LV\_LED\_BRIGHT\_MIN**

Brightness when the LED if OFF

**LV\_LED\_BRIGHT\_MAX**

Brightness when the LED if ON

**Functions****lv\_obj\_t \*lv\_led\_create(lv\_obj\_t \*parent)**

Create a led object

**Parametri**

- **parent** -- pointer to an object, it will be the parent of the new led

**Ritorna**

- pointer to the created led

**void lv\_led\_set\_color(lv\_obj\_t \*led, lv\_color\_t color)**

Set the color of the LED

**Parametri**

- **led** -- pointer to a LED object
- **color** -- the color of the LED

**void lv\_led\_set\_brightness(lv\_obj\_t \*led, uint8\_t bright)**

Set the brightness of a LED object

**Parametri**

- **led** -- pointer to a LED object
- **bright** -- LV\_LED\_BRIGHT\_MIN (max. dark) ... LV\_LED\_BRIGHT\_MAX (max. light)

```
void lv_led_on(lv_obj_t *led)
    Light on a LED
    Parametri
        led -- pointer to a LED object

void lv_led_off(lv_obj_t *led)
    Light off a LED
    Parametri
        led -- pointer to a LED object

void lv_led_toggle(lv_obj_t *led)
    Toggle the state of a LED
    Parametri
        led -- pointer to a LED object

uint8_t lv_led_get_brightness(const lv_obj_t *obj)
    Get the brightness of a LED object
    Parametri
        obj -- pointer to LED object
    Ritorna
        bright 0 (max. dark) ... 255 (max. light)
```

## Variables

const lv\_obj\_class\_t **lv\_led\_class**

## lv\_led\_private.h

```
struct _lv_led_t
#include <lv_led_private.h> Data of led
```

### Public Members

```
lv_obj_t obj  

lv_color_t color  

uint8_t bright  

    Current brightness of the LED (0..255)
```

## 14.24.18 line

### lv\_line.h

#### Functions

```
lv_obj_t *lv_line_create(lv_obj_t *parent)
    Create a line object
    Parametri
        parent -- pointer to an object, it will be the parent of the new line
    Ritorna
        pointer to the created line
```

```
void lv_line_set_points(lv_obj_t *obj, const lv_point_precise_t points[], uint32_t point_num)
    Set an array of points. The line object will connect these points.
    Parametri
```

- **obj** -- pointer to a line object
- **points** -- an array of points. Only the address is saved, so the array needs to be alive while the line exists
- **point\_num** -- number of points in 'point\_a'

**void lv\_line\_set\_points mutable(lv\_obj\_t \*obj, lv\_point\_precise\_t points[], uint32\_t point\_num)**

Set a non-const array of points. Identical to `lv_line_set_points` except the array may be retrieved by `lv_line_get_points mutable`.

**Parametri**

- **obj** -- pointer to a line object
- **points** -- a non-const array of points. Only the address is saved, so the array needs to be alive while the line exists.
- **point\_num** -- number of points in 'point\_a'

**void lv\_line\_set\_y\_invert(lv\_obj\_t \*obj, bool en)**

Enable (or disable) the y coordinate inversion. If enabled then y will be subtracted from the height of the object, therefore the y = 0 coordinate will be on the bottom.

**Parametri**

- **obj** -- pointer to a line object
- **en** -- true: enable the y inversion, false:disable the y inversion

**const lv\_point\_precise\_t \*lv\_line\_get\_points(lv\_obj\_t \*obj)**

Get the pointer to the array of points.

**Parametri**

**obj** -- pointer to a line object

**Ritorna**

const pointer to the array of points

**uint32\_t lv\_line\_get\_point\_count(lv\_obj\_t \*obj)**

Get the number of points in the array of points.

**Parametri**

**obj** -- pointer to a line object

**Ritorna**

number of points in array of points

**bool lv\_line\_is\_point\_array mutable(lv\_obj\_t \*obj)**

Check the mutability of the stored point array pointer.

**Parametri**

**obj** -- pointer to a line object

**Ritorna**

true: the point array pointer is mutable, false: constant

**lv\_point\_precise\_t \*lv\_line\_get\_points mutable(lv\_obj\_t \*obj)**

Get a pointer to the mutable array of points or NULL if it is not mutable

**Parametri**

**obj** -- pointer to a line object

**Ritorna**

pointer to the array of points. NULL if not mutable.

**bool lv\_line\_get\_y\_invert(const lv\_obj\_t \*obj)**

Get the y inversion attribute

**Parametri**

**obj** -- pointer to a line object

**Ritorna**

true: y inversion is enabled, false: disabled

## Variables

const lv\_obj\_class\_t **lv\_line\_class**

### lv\_line\_private.h

struct **\_lv\_line\_t**

#include <lv\_line\_private.h> Data of line

#### Public Members

lv\_obj\_t **obj**

const *lv\_point\_precise\_t* \***constant**

*lv\_point\_precise\_t* \***mut**

union *\_lv\_line\_t* **point\_array**

Pointer to an array with the points of the line

uint32\_t **point\_num**

Number of points in 'point\_array'

uint32\_t **y\_inv**

1: y == 0 will be on the bottom

uint32\_t **point\_array\_is Mutable**

whether the point array is const or mutable

## 14.24.19 list

### lv\_list.h

#### Functions

lv\_obj\_t \***lv\_list\_create**(lv\_obj\_t \*parent)

Create a list object

##### Parametri

- **parent** -- pointer to an object, it will be the parent of the new list

##### Ritorna

pointer to the created list

lv\_obj\_t \***lv\_list\_add\_text**(lv\_obj\_t \*list, const char \*txt)

Add text to a list

##### Parametri

- **list** -- pointer to a list, it will be the parent of the new label
- **txt** -- text of the new label

##### Ritorna

pointer to the created label

lv\_obj\_t \***lv\_list\_add\_button**(lv\_obj\_t \*list, const void \*icon, const char \*txt)

Add button to a list

##### Parametri

- **list** -- pointer to a list, it will be the parent of the new button

- **icon** -- icon for the button, when NULL it will have no icon
- **txt** -- text of the new button, when NULL no text will be added

**Ritorna**

pointer to the created button

```
const char *lv_list_get_button_text(lv_obj_t *list, lv_obj_t *btn)
```

Get text of a given list button

**Parametri**

- **list** -- pointer to a list
- **btn** -- pointer to the button

**Ritorna**

text of btn, if btn doesn't have text "" will be returned

```
void lv_list_set_button_text(lv_obj_t *list, lv_obj_t *btn, const char *txt)
```

Set text of a given list button

**Parametri**

- **list** -- pointer to a list
- **btn** -- pointer to the button
- **txt** -- pointer to the text

**Variables**

```
const lv_obj_class_t lv_list_class
```

```
const lv_obj_class_t lv_list_text_class
```

```
const lv_obj_class_t lv_list_button_class
```

**14.24.20 lottie**

**lv\_lottie.h**

**Functions**

```
lv_obj_t *lv_lottie_create(lv_obj_t *parent)
```

Create a lottie animation

**Parametri**

- **parent** -- pointer to the parent widget

**Ritorna**

pointer to the created Lottie animation widget

```
void lv_lottie_set_buffer(lv_obj_t *obj, int32_t w, int32_t h, void *buf)
```

Set a buffer for the animation. It also defines the size of the animation

**Parametri**

- **obj** -- pointer to a lottie widget
- **w** -- width of the animation and buffer
- **h** -- height of the animation and buffer
- **buf** -- a static buffer with **width** x **height** x 4 byte size

```
void lv_lottie_set_draw_buf(lv_obj_t *obj, lv_draw_buf_t *draw_buf)
```

Set a draw buffer for the animation. It also defines the size of the animation

**Parametri**

- **obj** -- pointer to a lottie widget
- **draw\_buf** -- an initialized draw buffer with ARGB8888 color format

```
void lv_lottie_set_src_data(lv_obj_t *obj, const void *src, size_t src_size)
```

Set the source for the animation as an array

**Parametri**

- **obj** -- pointer to a lottie widget
- **src** -- the lottie animation converted to an nul terminated array
- **src\_size** -- size of the source array in bytes

**void lv\_lottie\_set\_src\_file(lv\_obj\_t \*obj, const char \*src)**

Set the source for the animation as a path. Lottie doesn't use LVGL's File System API.

**Parametri**

- **obj** -- pointer to a lottie widget
- **src** -- path to a json file, e.g. "path/to/file.json"

**lv\_anim\_t \*lv\_lottie\_get\_anim(lv\_obj\_t \*obj)**

Get the LVGL animation which controls the lottie animation

**Parametri**

- **obj** -- pointer to a lottie widget

**Ritorna**

the LVGL animation

**lv\_lottie\_private.h**

**struct lv\_lottie\_t**

**Public Members**

lv\_canvas\_t **canvas**

Tvg\_Paint \***tvg\_paint**

Tvg\_Canvas \***tvg\_canvas**

Tvg\_Animation \***tvg\_anim**

lv\_anim\_t \***anim**

int32\_t **last\_rendered\_time**

## 14.24.21 menu

**lv\_menu.h**

**Enums**

enum **lv\_menu\_mode\_header\_t**

*Values:*

enumerator **LV\_MENU\_HEADER\_TOP\_FIXED**

Header is positioned at the top

enumerator **LV\_MENU\_HEADER\_TOP\_UNFIXED**

Header is positioned at the top and can be scrolled out of view

enumerator **LV\_MENU\_HEADER\_BOTTOM\_FIXED**

Header is positioned at the bottom

---

enum **lv\_menu\_mode\_root\_back\_button\_t**

**Values:**

enumerator **LV\_MENU\_ROOT\_BACK\_BUTTON\_DISABLED**

enumerator **LV\_MENU\_ROOT\_BACK\_BUTTON\_ENABLED**

## Functions

lv\_obj\_t \***lv\_menu\_create**(lv\_obj\_t \*parent)

Create a menu object

**Parametri**

**parent** -- pointer to an object, it will be the parent of the new menu

**Ritorna**

pointer to the created menu

lv\_obj\_t \***lv\_menu\_page\_create**(lv\_obj\_t \*menu, char const \*const title)

Create a menu page object.

This call inserts the new page under menu->storage as its parent, which is itself a child of the menu, so the resulting object hierarchy is: menu => storage => new\_page where **storage** is a Base Widget.

**Parametri**

- **menu** -- pointer to menu object.
- **title** -- pointer to text for title in header (NULL to not display title)

**Ritorna**

pointer to the created menu page

lv\_obj\_t \***lv\_menu\_cont\_create**(lv\_obj\_t \*parent)

Create a menu cont object

**Parametri**

**parent** -- pointer to a menu page object, it will be the parent of the new menu cont object

**Ritorna**

pointer to the created menu cont

lv\_obj\_t \***lv\_menu\_section\_create**(lv\_obj\_t \*parent)

Create a menu section object

**Parametri**

**parent** -- pointer to a menu page object, it will be the parent of the new menu section object

**Ritorna**

pointer to the created menu section

lv\_obj\_t \***lv\_menu\_separator\_create**(lv\_obj\_t \*parent)

Create a menu separator object

**Parametri**

**parent** -- pointer to a menu page object, it will be the parent of the new menu separator object

**Ritorna**

pointer to the created menu separator

void **lv\_menu\_set\_page**(lv\_obj\_t \*obj, lv\_obj\_t \*page)

Set menu page to display in main

**Parametri**

- **obj** -- pointer to the menu
- **page** -- pointer to the menu page to set (NULL to clear main and clear menu history)

void **lv\_menu\_set\_page\_title**(lv\_obj\_t \*page, char const \*const title)

Set menu page title

**Parametri**

- **page** -- pointer to the menu page
- **title** -- pointer to text for title in header (NULL to not display title)

**void lv\_menu\_set\_page\_title\_static(lv\_obj\_t \*page, char const \*const title)**

Set menu page title with a static text. It will not be saved by the label so the 'text' variable has to be 'alive' while the page exists.

**Parametri**

- **page** -- pointer to the menu page
- **title** -- pointer to text for title in header (NULL to not display title)

**void lv\_menu\_set\_sidebar\_page(lv\_obj\_t \*obj, lv\_obj\_t \*page)**

Set menu page to display in sidebar

**Parametri**

- **obj** -- pointer to the menu
- **page** -- pointer to the menu page to set (NULL to clear sidebar)

**void lv\_menu\_set\_mode\_header(lv\_obj\_t \*obj, lv\_menu\_mode\_header\_t mode)**

Set the how the header should behave and its position

**Parametri**

- **obj** -- pointer to a menu
- **mode** -- LV\_MENU\_HEADER\_TOP\_FIXED/TOP\_UNFIXED/BOTTOM\_FIXED

**void lv\_menu\_set\_mode\_root\_back\_button(lv\_obj\_t \*obj, lv\_menu\_mode\_root\_back\_button\_t mode)**

Set whether back button should appear at root

**Parametri**

- **obj** -- pointer to a menu
- **mode** -- LV\_MENU\_ROOT\_BACK\_BUTTON\_DISABLED/ENABLED

**void lv\_menu\_set\_load\_page\_event(lv\_obj\_t \*menu, lv\_obj\_t \*obj, lv\_obj\_t \*page)**

Add menu to the menu item

**Parametri**

- **menu** -- pointer to the menu
- **obj** -- pointer to the obj
- **page** -- pointer to the page to load when obj is clicked

**lv\_obj\_t \*lv\_menu\_get\_cur\_main\_page(lv\_obj\_t \*obj)**

Get a pointer to menu page that is currently displayed in main

**Parametri**

- **obj** -- pointer to the menu

**Ritorna**

pointer to current page

**lv\_obj\_t \*lv\_menu\_get\_cur\_sidebar\_page(lv\_obj\_t \*obj)**

Get a pointer to menu page that is currently displayed in sidebar

**Parametri**

- **obj** -- pointer to the menu

**Ritorna**

pointer to current page

**lv\_obj\_t \*lv\_menu\_get\_main\_header(lv\_obj\_t \*obj)**

Get a pointer to main header obj

**Parametri**

- **obj** -- pointer to the menu

**Ritorna**

pointer to main header obj

**lv\_obj\_t \*lv\_menu\_get\_main\_header\_back\_button(lv\_obj\_t \*obj)**

Get a pointer to main header back btn obj

**Parametri**  
**obj** -- pointer to the menu  
**Ritorna**  
pointer to main header back btn obj

**lv\_obj\_t \*lv\_menu\_get\_sidebar\_header(lv\_obj\_t \*obj)**  
Get a pointer to sidebar header obj  
**Parametri**  
**obj** -- pointer to the menu  
**Ritorna**  
pointer to sidebar header obj

**lv\_obj\_t \*lv\_menu\_get\_sidebar\_header\_back\_button(lv\_obj\_t \*obj)**  
Get a pointer to sidebar header obj  
**Parametri**  
**obj** -- pointer to the menu  
**Ritorna**  
pointer to sidebar header back btn obj

**bool lv\_menu\_back\_button\_is\_root(lv\_obj\_t \*menu, lv\_obj\_t \*obj)**  
Check if an obj is a root back btn  
**Parametri**  
• **menu** -- pointer to the menu  
• **obj** -- pointer to the back button  
**Ritorna**  
true if it is a root back btn

**void lv\_menu\_clear\_history(lv\_obj\_t \*obj)**  
Clear menu history  
**Parametri**  
**obj** -- pointer to the menu

## Variables

```
const lv_obj_class_t lv_menu_class

const lv_obj_class_t lv_menu_page_class

const lv_obj_class_t lv_menu_cont_class

const lv_obj_class_t lv_menu_section_class

const lv_obj_class_t lv_menu_separator_class

const lv_obj_class_t lv_menu_sidebar_cont_class

const lv_obj_class_t lv_menu_main_cont_class

const lv_obj_class_t lv_menu_sidebar_header_cont_class

const lv_obj_class_t lv_menu_main_header_cont_class
```

**lv\_menu\_private.h**

```
struct _lv_menu_load_page_event_data_t
```

**Public Members**

lv\_obj\_t \*menu

lv\_obj\_t \*page

**struct \_lv\_menu\_history\_t****Public Members**

lv\_obj\_t \*page

**struct \_lv\_menu\_t****Public Members**

lv\_obj\_t obj

lv\_obj\_t \*storage

a pointer to obj that is the parent of all pages not displayed

lv\_obj\_t \*main

lv\_obj\_t \*main\_page

lv\_obj\_t \*main\_header

lv\_obj\_t \*main\_header\_back\_btn

a pointer to obj that on click triggers back btn event handler, can be same as 'main\_header'

lv\_obj\_t \*main\_header\_title

lv\_obj\_t \*sidebar

lv\_obj\_t \*sidebar\_page

lv\_obj\_t \*sidebar\_header

lv\_obj\_t \*sidebar\_header\_back\_btn

a pointer to obj that on click triggers back btn event handler, can be same as 'sidebar\_header'

lv\_obj\_t \*sidebar\_header\_title

```

lv_obj_t *selected_tab

lv_ll_t history_ll

uint8_t cur_depth

uint8_t prev_depth

uint8_t sidebar_generated

lv_menu_mode_header_t mode_header

lv_menu_mode_root_back_button_t mode_root_back_btn

struct _lv_menu_page_t

```

**Public Members**

```

lv_obj_t obj

char *title

bool static_title

```

**14.24.22 msgbox****lv\_msgbox.h****Functions****lv\_obj\_t \*lv\_msgbox\_create(lv\_obj\_t \*parent)**

Create an empty message box

**Parametri**

parent -- the parent or NULL to create a modal msgbox

**Ritorna**

the created message box

**lv\_obj\_t \*lv\_msgbox\_add\_title(lv\_obj\_t \*obj, const char \*title)**

Add title to the message box. It also creates a header for the title.

**Parametri**

- **obj** -- pointer to a message box
- **title** -- the text of the tile

**Ritorna**

the created title label

**lv\_obj\_t \*lv\_msgbox\_add\_header\_button(lv\_obj\_t \*obj, const void \*icon)**

Add a button to the header of the message box. It also creates a header.

**Parametri**

- **obj** -- pointer to a message box
- **icon** -- the icon of the button

**Ritorna**

the created button

`lv_obj_t *lv_msgbox_add_text(lv_obj_t *obj, const char *text)`

Add a text to the content area of message box. Multiple texts will be created below each other.

**Parametri**

- **obj** -- pointer to a message box
- **text** -- text to add

**Ritorna**

the created button

`lv_obj_t *lv_msgbox_add_footer_button(lv_obj_t *obj, const char *text)`

Add a button to the footer of to the message box. It also creates a footer.

**Parametri**

- **obj** -- pointer to a message box
- **text** -- the text of the button

**Ritorna**

the created button

`lv_obj_t *lv_msgbox_add_close_button(lv_obj_t *obj)`

Add a close button to the message box. It also creates a header.

**Parametri**

**obj** -- pointer to a message box

**Ritorna**

the created close button

`lv_obj_t *lv_msgbox_get_header(lv_obj_t *obj)`

Get the header widget

**Parametri**

**obj** -- pointer to a message box

**Ritorna**

the header, or NULL if not exists

`lv_obj_t *lv_msgbox_get_footer(lv_obj_t *obj)`

Get the footer widget

**Parametri**

**obj** -- pointer to a message box

**Ritorna**

the footer, or NULL if not exists

`lv_obj_t *lv_msgbox_get_content(lv_obj_t *obj)`

Get the content widget

**Parametri**

**obj** -- pointer to a message box

**Ritorna**

the content

`lv_obj_t *lv_msgbox_get_title(lv_obj_t *obj)`

Get the title label

**Parametri**

**obj** -- pointer to a message box

**Ritorna**

the title, or NULL if it does not exist

`void lv_msgbox_close(lv_obj_t *mbox)`

Close a message box

**Parametri**

**mbox** -- pointer to a message box

`void lv_msgbox_close_async(lv_obj_t *mbox)`

Close a message box in the next call of the message box

**Parametri**

**mbox** -- pointer to a message box

**Variables**

```
const lv_obj_class_t lv_msgbox_class  

const lv_obj_class_t lv_msgbox_header_class  

const lv_obj_class_t lv_msgbox_content_class  

const lv_obj_class_t lv_msgbox_footer_class  

const lv_obj_class_t lv_msgbox_header_button_class  

const lv_obj_class_t lv_msgbox_footer_button_class  

const lv_obj_class_t lv_msgbox_backdrop_class  

lv_msgbox_private.h  

struct _lv_msgbox_t
```

**Public Members**

```
lv_obj_t obj  

lv_obj_t *header  

lv_obj_t *content  

lv_obj_t *footer  

lv_obj_t *title
```

**14.24.23 objx\_tmpl**

**lv\_objx\_tmpl.h**

**14.24.24 property**

**lv\_obj\_property\_names.h**

GENERATED FILE, DO NOT EDIT IT!

**Variables**

```
const lv_property_name_t lv_animimage_property_names[4]  

const lv_property_name_t lv_dropdown_property_names[9]  

const lv_property_name_t lv_image_property_names[11]
```

```
const lv_property_name_t lv_keyboard_property_names[4]
```

```
const lv_property_name_t lv_label_property_names[4]
```

```
const lv_property_name_t lv_obj_property_names[75]
```

```
const lv_property_name_t lv_roller_property_names[3]
```

```
const lv_property_name_t lv_slider_property_names[8]
```

```
const lv_property_name_t lv_style_property_names[120]
```

```
const lv_property_name_t lv_textarea_property_names[15]
```

## **lv\_style\_properties.h**

GENERATED FILE, DO NOT EDIT IT!

### Enums

```
enum _lv_property_style_id_t
```

*Values:*

```
enumerator LV_PROPERTY_STYLE_ALIGN
```

```
enumerator LV_PROPERTY_STYLE_ANIM
```

```
enumerator LV_PROPERTY_STYLE_ANIM_DURATION
```

```
enumerator LV_PROPERTY_STYLE_ARC_COLOR
```

```
enumerator LV_PROPERTY_STYLE_ARC_IMAGE_SRC
```

```
enumerator LV_PROPERTY_STYLE_ARC_OPA
```

```
enumerator LV_PROPERTY_STYLE_ARC_ROUNDED
```

```
enumerator LV_PROPERTY_STYLE_ARC_WIDTH
```

```
enumerator LV_PROPERTY_STYLE_BASE_DIR
```

```
enumerator LV_PROPERTY_STYLE_BG_COLOR
```

```
enumerator LV_PROPERTY_STYLE_BG_GRAD
```

```
enumerator LV_PROPERTY_STYLE_BG_GRAD_COLOR
```

enumerator **LV\_PROPERTY\_STYLE\_BG\_GRAD\_DIR**

enumerator **LV\_PROPERTY\_STYLE\_BG\_GRAD\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_BG\_GRAD\_STOP**

enumerator **LV\_PROPERTY\_STYLE\_BG\_IMAGE\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_BG\_IMAGE\_RECOLOR**

enumerator **LV\_PROPERTY\_STYLE\_BG\_IMAGE\_RECOLOR\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_BG\_IMAGE\_SRC**

enumerator **LV\_PROPERTY\_STYLE\_BG\_IMAGE\_TILED**

enumerator **LV\_PROPERTY\_STYLE\_BG\_MAIN\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_BG\_MAIN\_STOP**

enumerator **LV\_PROPERTY\_STYLE\_BG\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_BITMAP\_MASK\_SRC**

enumerator **LV\_PROPERTY\_STYLE\_BLEND\_MODE**

enumerator **LV\_PROPERTY\_STYLE\_BORDER\_COLOR**

enumerator **LV\_PROPERTY\_STYLE\_BORDER\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_BORDER\_POST**

enumerator **LV\_PROPERTY\_STYLE\_BORDER\_SIDE**

enumerator **LV\_PROPERTY\_STYLE\_BORDER\_WIDTH**

enumerator **LV\_PROPERTY\_STYLE\_CLIP\_CORNER**

enumerator **LV\_PROPERTY\_STYLE\_COLOR\_FILTER\_DSC**

enumerator **LV\_PROPERTY\_STYLE\_COLOR\_FILTER\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_FLEX\_CROSS\_PLACE**

enumerator **LV\_PROPERTY\_STYLE\_FLEX\_FLOW**

enumerator **LV\_PROPERTY\_STYLE\_FLEX\_GROW**

enumerator **LV\_PROPERTY\_STYLE\_FLEX\_MAIN\_PLACE**

enumerator **LV\_PROPERTY\_STYLE\_FLEX\_TRACK\_PLACE**

enumerator **LV\_PROPERTY\_STYLE\_GRID\_CELL\_COLUMN\_POS**

enumerator **LV\_PROPERTY\_STYLE\_GRID\_CELL\_COLUMN\_SPAN**

enumerator **LV\_PROPERTY\_STYLE\_GRID\_CELL\_ROW\_POS**

enumerator **LV\_PROPERTY\_STYLE\_GRID\_CELL\_ROW\_SPAN**

enumerator **LV\_PROPERTY\_STYLE\_GRID\_CELL\_X\_ALIGN**

enumerator **LV\_PROPERTY\_STYLE\_GRID\_CELL\_Y\_ALIGN**

enumerator **LV\_PROPERTY\_STYLE\_GRID\_COLUMN\_ALIGN**

enumerator **LV\_PROPERTY\_STYLE\_GRID\_COLUMN\_DSC\_ARRAY**

enumerator **LV\_PROPERTY\_STYLE\_GRID\_ROW\_ALIGN**

enumerator **LV\_PROPERTY\_STYLE\_GRID\_ROW\_DSC\_ARRAY**

enumerator **LV\_PROPERTY\_STYLE\_HEIGHT**

enumerator **LV\_PROPERTY\_STYLE\_IMAGE\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_IMAGE\_RECOLOR**

enumerator **LV\_PROPERTY\_STYLE\_IMAGE\_RECOLOR\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_LAST\_BUILT\_IN\_PROP**

enumerator **LV\_PROPERTY\_STYLE\_LAYOUT**

enumerator **LV\_PROPERTY\_STYLE\_LENGTH**

enumerator **LV\_PROPERTY\_STYLE\_LINE\_COLOR**

enumerator **LV\_PROPERTY\_STYLE\_LINE\_DASH\_GAP**

enumerator **LV\_PROPERTY\_STYLE\_LINE\_DASH\_WIDTH**

enumerator **LV\_PROPERTY\_STYLE\_LINE\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_LINE\_ROUNDED**

enumerator **LV\_PROPERTY\_STYLE\_LINE\_WIDTH**

enumerator **LV\_PROPERTY\_STYLE\_MARGIN\_BOTTOM**

enumerator **LV\_PROPERTY\_STYLE\_MARGIN\_LEFT**

enumerator **LV\_PROPERTY\_STYLE\_MARGIN\_RIGHT**

enumerator **LV\_PROPERTY\_STYLE\_MARGIN\_TOP**

enumerator **LV\_PROPERTY\_STYLE\_MAX\_HEIGHT**

enumerator **LV\_PROPERTY\_STYLE\_MAX\_WIDTH**

enumerator **LV\_PROPERTY\_STYLE\_MIN\_HEIGHT**

enumerator **LV\_PROPERTY\_STYLE\_MIN\_WIDTH**

enumerator **LV\_PROPERTY\_STYLE\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_OPA\_LAYERED**

enumerator **LV\_PROPERTY\_STYLE\_OUTLINE\_COLOR**

enumerator **LV\_PROPERTY\_STYLE\_OUTLINE\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_OUTLINE\_PAD**

enumerator **LV\_PROPERTY\_STYLE\_OUTLINE\_WIDTH**

enumerator **LV\_PROPERTY\_STYLE\_PAD\_BOTTOM**

enumerator **LV\_PROPERTY\_STYLE\_PAD\_COLUMN**

enumerator **LV\_PROPERTY\_STYLE\_PAD\_LEFT**

enumerator **LV\_PROPERTY\_STYLE\_PAD\_RADIAL**

enumerator **LV\_PROPERTY\_STYLE\_PAD\_RIGHT**

enumerator **LV\_PROPERTY\_STYLE\_PAD\_ROW**

enumerator **LV\_PROPERTY\_STYLE\_PAD\_TOP**

enumerator **LV\_PROPERTY\_STYLE\_PROP\_INV**

enumerator **LV\_PROPERTY\_STYLE\_RADIAL\_OFFSET**

enumerator **LV\_PROPERTY\_STYLE\_RADIUS**

enumerator **LV\_PROPERTY\_STYLE\_RECOLOR**

enumerator **LV\_PROPERTY\_STYLE\_RECOLOR\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_ROTARY\_SENSITIVITY**

enumerator **LV\_PROPERTY\_STYLE\_SHADOW\_COLOR**

enumerator **LV\_PROPERTY\_STYLE\_SHADOW\_OFFSET\_X**

enumerator **LV\_PROPERTY\_STYLE\_SHADOW\_OFFSET\_Y**

enumerator **LV\_PROPERTY\_STYLE\_SHADOW\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_SHADOW\_SPREAD**

enumerator **LV\_PROPERTY\_STYLE\_SHADOW\_WIDTH**

enumerator **LV\_PROPERTY\_STYLE\_TEXT\_ALIGN**

enumerator **LV\_PROPERTY\_STYLE\_TEXT\_COLOR**

enumerator **LV\_PROPERTY\_STYLE\_TEXT\_DECOR**

enumerator **LV\_PROPERTY\_STYLE\_TEXT\_FONT**

enumerator **LV\_PROPERTY\_STYLE\_TEXT\_LETTER\_SPACE**

enumerator **LV\_PROPERTY\_STYLE\_TEXT\_LINE\_SPACE**

enumerator **LV\_PROPERTY\_STYLE\_TEXT\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_TEXT\_OUTLINE\_STROKE\_COLOR**

enumerator **LV\_PROPERTY\_STYLE\_TEXT\_OUTLINE\_STROKE\_OPA**

enumerator **LV\_PROPERTY\_STYLE\_TEXT\_OUTLINE\_STROKE\_WIDTH**

enumerator **LV\_PROPERTY\_STYLE\_TRANSFORM\_HEIGHT**

enumerator **LV\_PROPERTY\_STYLE\_TRANSFORM\_PIVOT\_X**

enumerator **LV\_PROPERTY\_STYLE\_TRANSFORM\_PIVOT\_Y**

enumerator **LV\_PROPERTY\_STYLE\_TRANSFORM\_ROTATION**

enumerator **LV\_PROPERTY\_STYLE\_TRANSFORM\_SCALE\_X**

enumerator **LV\_PROPERTY\_STYLE\_TRANSFORM\_SCALE\_Y**

enumerator **LV\_PROPERTY\_STYLE\_TRANSFORM\_SKew\_X**

enumerator **LV\_PROPERTY\_STYLE\_TRANSFORM\_SKew\_Y**

enumerator **LV\_PROPERTY\_STYLE\_TRANSFORM\_WIDTH**

enumerator **LV\_PROPERTY\_STYLE\_TRANSITION**

enumerator **LV\_PROPERTY\_STYLE\_TRANSLATE\_RADIAL**

enumerator **LV\_PROPERTY\_STYLE\_TRANSLATE\_X**

enumerator **LV\_PROPERTY\_STYLE\_TRANSLATE\_Y**

enumerator **LV\_PROPERTY\_STYLE\_WIDTH**

enumerator **LV\_PROPERTY\_STYLE\_X**

enumerator **LV\_PROPERTY\_STYLE\_Y**

## 14.24.25 roller

[lv\\_roller.h](#)

### Enums

enum **lv\_roller\_mode\_t**

Roller mode.

*Values:*

enumerator **LV\_ROLLER\_MODE\_NORMAL**

Normal mode (roller ends at the end of the options).

enumerator **LV\_ROLLER\_MODE\_INFINITE**

Infinite mode (roller can be scrolled forever).

---

enum **\_lv\_property\_roller\_id\_t**

*Values:*

enumerator **LV\_PROPERTY\_ROLLER\_OPTIONS**

enumerator **LV\_PROPERTY\_ROLLER\_SELECTED**

enumerator **LV\_PROPERTY\_ROLLER\_VISIBLE\_ROW\_COUNT**

enumerator **LV\_PROPERTY\_ROLLER\_END**

## Functions

**lv\_obj\_t \*lv\_roller\_create(lv\_obj\_t \*parent)**

Create a roller object

**Parametri**

- **parent** -- pointer to an object, it will be the parent of the new roller.

**Ritorna**

- pointer to the created roller

**void lv\_roller\_set\_options(lv\_obj\_t \*obj, const char \*options, lv\_roller\_mode\_t mode)**

Set the options on a roller

**Parametri**

- **obj** -- pointer to roller object
- **options** -- a string with '  
' separated options. E.g. "One\nTwo\nThree"
- **mode** -- LV\_ROLLER\_MODE\_NORMAL or LV\_ROLLER\_MODE\_INFINITE

**void lv\_roller\_set\_selected(lv\_obj\_t \*obj, uint32\_t sel\_opt, lv\_anim\_enable\_t anim)**

Set the selected option

**Parametri**

- **obj** -- pointer to a roller object
- **sel\_opt** -- index of the selected option (0 ... number of option - 1);
- **anim** -- LV\_ANIM\_ON: set with animation; LV\_ANIM\_OFF set immediately

**bool lv\_roller\_set\_selected\_str(lv\_obj\_t \*obj, const char \*sel\_opt, lv\_anim\_enable\_t anim)**

Sets the given string as the selection on the roller. Does not alter the current selection on failure.

**Parametri**

- **obj** -- pointer to roller object
- **sel\_opt** -- pointer to the string you want to set as an option
- **anim** -- LV\_ANIM\_ON: set with animation; LV\_ANIM\_OFF set immediately

**Ritorna**

- true** if set successfully and **false** if the given string does not exist as an option in the roller

**void lv\_roller\_set\_visible\_row\_count(lv\_obj\_t \*obj, uint32\_t row\_cnt)**

Set the height to show the given number of rows (options)

**Parametri**

- **obj** -- pointer to a roller object
- **row\_cnt** -- number of desired visible rows

**uint32\_t lv\_roller\_get\_selected(const lv\_obj\_t \*obj)**

Get the index of the selected option

**Parametri**

- obj** -- pointer to a roller object

**Ritorna**

index of the selected option (0 ... number of option - 1);

**void lv\_roller\_get\_selected\_str(const lv\_obj\_t \*obj, char \*buf, uint32\_t buf\_size)**

Get the current selected option as a string.

**Parametri**

- **obj** -- pointer to roller object
- **buf** -- pointer to an array to store the string
- **buf\_size** -- size of **buf** in bytes. 0: to ignore it.

**const char \*lv\_roller\_get\_options(const lv\_obj\_t \*obj)**

Get the options of a roller

**Parametri**

**obj** -- pointer to roller object

**Ritorna**

the options separated by '

'-s (E.g. "Option1\nOption2\nOption3")

**uint32\_t lv\_roller\_get\_option\_count(const lv\_obj\_t \*obj)**

Get the total number of options

**Parametri**

**obj** -- pointer to a roller object

**Ritorna**

the total number of options

**Variables**

**const lv\_obj\_class\_t lv\_roller\_class**

**lv\_roller\_private.h**

**struct \_lv\_roller\_t**

**Public Members**

**lv\_obj\_t obj**

**uint32\_t option\_cnt**

Number of options

**uint32\_t sel\_opt\_id**

Index of the current option

**uint32\_t sel\_opt\_id\_ori**

Store the original index on focus

**uint32\_t inf\_page\_cnt**

Number of extra pages added to make the roller look infinite

**lv\_roller\_mode\_t mode**

**uint32\_t moved**

## 14.24.26 scale

`lv_scale.h`

### Defines

**LV\_SCALE\_TOTAL\_TICK\_COUNT\_DEFAULT**

Default value of total minor ticks.

**LV\_SCALE\_MAJOR\_TICK\_EVERY\_DEFAULT**

Default value of major tick every nth ticks.

**LV\_SCALE\_LABEL\_ENABLED\_DEFAULT**

Default value of scale label enabled.

**LV\_SCALE\_LABEL\_ROTATE\_MATCH\_TICKS**

**LV\_SCALE\_LABEL\_ROTATE\_KEEP\_UPRIGHT**

**LV\_SCALE\_ROTATION\_ANGLE\_MASK**

### Enums

enum **lv\_scale\_mode\_t**

Scale mode

*Values:*

enumerator **LV\_SCALE\_MODE\_HORIZONTAL\_TOP**

enumerator **LV\_SCALE\_MODE\_HORIZONTAL\_BOTTOM**

enumerator **LV\_SCALE\_MODE\_VERTICAL\_LEFT**

enumerator **LV\_SCALE\_MODE\_VERTICAL\_RIGHT**

enumerator **LV\_SCALE\_MODE\_ROUND\_INNER**

enumerator **LV\_SCALE\_MODE\_ROUND\_OUTER**

enumerator **LV\_SCALE\_MODE\_LAST**

### Functions

`lv_obj_t *lv_scale_create(lv_obj_t *parent)`

Create an scale object

#### Parametri

**parent** -- pointer to an object, it will be the parent of the new scale

#### Ritorna

pointer to created Scale Widget

**void lv\_scale\_set\_mode(lv\_obj\_t \*obj, *lv\_scale\_mode\_t* mode)**

Set scale mode. See [lv\\_scale\\_mode\\_t](#).

**Parametri**

- **obj** -- pointer to Scale Widget
- **mode** -- the new scale mode

**void lv\_scale\_set\_total\_tick\_count(lv\_obj\_t \*obj, uint32\_t total\_tick\_count)**

Set scale total tick count (including minor and major ticks).

**Parametri**

- **obj** -- pointer to Scale Widget
- **total\_tick\_count** -- New total tick count

**void lv\_scale\_set\_major\_tick\_every(lv\_obj\_t \*obj, uint32\_t major\_tick\_every)**

Sets how often major ticks are drawn.

**Parametri**

- **obj** -- pointer to Scale Widget
- **major\_tick\_every** -- the new count for major tick drawing

**void lv\_scale\_set\_label\_show(lv\_obj\_t \*obj, bool show\_label)**

Sets label visibility.

**Parametri**

- **obj** -- pointer to Scale Widget
- **show\_label** -- true/false to enable tick label

**void lv\_scale\_set\_range(lv\_obj\_t \*obj, int32\_t min, int32\_t max)**

Set minimum and maximum values on Scale.

**Parametri**

- **obj** -- pointer to Scale Widget
- **min** -- minimum value of Scale
- **max** -- maximum value of Scale

**void lv\_scale\_set\_min\_value(lv\_obj\_t \*obj, int32\_t min)**

Set minimum values on Scale.

**Parametri**

- **obj** -- pointer to Scale Widget
- **min** -- minimum value of Scale

**void lv\_scale\_set\_max\_value(lv\_obj\_t \*obj, int32\_t max)**

Set maximum values on Scale.

**Parametri**

- **obj** -- pointer to Scale Widget
- **min** -- minimum value of Scale

**void lv\_scale\_set\_angle\_range(lv\_obj\_t \*obj, uint32\_t angle\_range)**

Set angle between the low end and the high end of the Scale. (Applies only to round Scales.)

**Parametri**

- **obj** -- pointer to Scale Widget
- **angle\_range** -- angle in degrees from Scale minimum where top end of Scale will be drawn

**void lv\_scale\_set\_rotation(lv\_obj\_t \*obj, int32\_t rotation)**

Set angular offset from the 3-o'clock position of the low end of the Scale. (Applies only to round Scales.)

**Parametri**

- **obj** -- pointer to Scale Widget
- **rotation** -- clockwise angular offset (in degrees) from the 3-o'clock position of the low end of the scale; negative and >360 values are first normalized to range [0..360]. Examples:

- 0 = 3 o'clock (right side)
- 30 = 4 o'clock

- 60 = 5 o'clock
- 90 = 6 o'clock
- 135 = midway between 7 and 8 o'clock (default)
- 180 = 9 o'clock
- 270 = 12 o'clock
- 300 = 1 o'clock
- 330 = 2 o'clock
- -30 = 2 o'clock
- 390 = 4 o'clock

```
void lv_scale_set_line_needle_value(lv_obj_t *obj, lv_obj_t *needle_line, int32_t needle_length,
int32_t value)
```

Point line needle to specified value.

#### Parametri

- **obj** -- pointer to Scale Widget
- **needle\_line** -- needle\_line of the Scale. The line points will be allocated and managed by the Scale unless the line point array was previously set using `lv_line_set_points Mutable`.
- **needle\_length** -- length of the needle
  - needle\_length>0: needle\_length=needle\_length;
  - needle\_length<0: needle\_length=radius-|needle\_length|;
- **value** -- Scale value needle will point to

```
void lv_scale_set_image_needle_value(lv_obj_t *obj, lv_obj_t *needle_img, int32_t value)
```

Point image needle to specified value; image must point to the right. E.g. -O----->

#### Parametri

- **obj** -- pointer to Scale Widget
- **needle\_img** -- pointer to needle's Image
- **value** -- Scale value needle will point to

```
void lv_scale_set_text_src(lv_obj_t *obj, const char *txt_src[])
```

Set custom text source for major ticks labels.

#### Parametri

- **obj** -- pointer to Scale Widget
- **txt\_src** -- pointer to an array of strings which will be display at major ticks; last element must be a NULL pointer.

```
void lv_scale_set_post_draw(lv_obj_t *obj, bool en)
```

Draw Scale after all its children are drawn.

#### Parametri

- **obj** -- pointer to Scale Widget
- **en** -- true: enable post draw

```
void lv_scale_set_draw_ticks_on_top(lv_obj_t *obj, bool en)
```

Draw Scale ticks on top of all other parts.

#### Parametri

- **obj** -- pointer to Scale Widget
- **en** -- true: enable draw ticks on top of all parts

```
lv_scale_section_t *lv_scale_add_section(lv_obj_t *obj)
```

Add a Section to specified Scale. Section will not be drawn until a valid range is set for it using `lv_scale_set_section_range()`.

#### Parametri

- obj** -- pointer to Scale Widget

**Ritorna**

pointer to new Section

**void lv\_scale\_section\_set\_range(lv\_scale\_section\_t \*section, int32\_t min, int32\_t max)**

DEPRECATED, use lv\_scale\_set\_section\_range instead. Set range for specified Scale Section

**Parametri**

- **section** -- pointer to Section
- **range\_min** -- Section new minimum value
- **range\_max** -- Section new maximum value

**void lv\_scale\_set\_section\_range(lv\_obj\_t \*scale, lv\_scale\_section\_t \*section, int32\_t min, int32\_t max)**

Set the range of a scale section

**Parametri**

- **scale** -- pointer to scale
- **section** -- pointer to section
- **range\_min** -- the section's new minimum value
- **range\_max** -- the section's new maximum value

**void lv\_scale\_set\_section\_min\_value(lv\_obj\_t \*scale, lv\_scale\_section\_t \*section, int32\_t min)**

Set the minimum value of a scale section

**Parametri**

- **scale** -- pointer to scale
- **section** -- pointer to section
- **min** -- the section's new minimum value

**void lv\_scale\_set\_section\_max\_value(lv\_obj\_t \*scale, lv\_scale\_section\_t \*section, int32\_t max)**

Set the maximum value of a scale section

**Parametri**

- **scale** -- pointer to scale
- **section** -- pointer to section
- **max** -- the section's new maximum value

**void lv\_scale\_section\_set\_style(lv\_scale\_section\_t \*section, *lv\_part\_t* part, *lv\_style\_t* \*section\_part\_style)**

DEPRECATED, use lv\_scale\_set\_section\_style\_main/indicator/items instead. Set style for specified part of Section.

**Parametri**

- **section** -- pointer to Section
- **part** -- the part of the Scale the style will apply to, e.g. LV\_PART\_INDICATOR
- **section\_part\_style** -- pointer to style to apply

**void lv\_scale\_set\_section\_style\_main(lv\_obj\_t \*scale, lv\_scale\_section\_t \*section, const *lv\_style\_t* \*style)**

Set the style of the line on a section.

**Parametri**

- **scale** -- pointer to scale
- **section** -- pointer to section
- **style** -- point to a style

**void lv\_scale\_set\_section\_style\_indicator(lv\_obj\_t \*scale, lv\_scale\_section\_t \*section, const *lv\_style\_t* \*style)**

Set the style of the major ticks and label on a section.

**Parametri**

- **scale** -- pointer to scale
- **section** -- pointer to section
- **style** -- point to a style

**void lv\_scale\_set\_section\_style\_items(lv\_obj\_t \*scale, lv\_scale\_section\_t \*section, const *lv\_style\_t* \*style)**

Set the style of the minor ticks on a section.

**Parametri**

- **scale** -- pointer to scale
- **section** -- pointer to section
- **style** -- point to a style

`lv_scale_mode_t lv_scale_get_mode(lv_obj_t *obj)`

Get scale mode. See [lv\\_scale\\_mode\\_t](#)

**Parametri**

- **obj** -- pointer to Scale Widget

**Ritorna**

Scale mode

`int32_t lv_scale_get_total_tick_count(lv_obj_t *obj)`

Get scale total tick count (including minor and major ticks)

**Parametri**

- **obj** -- pointer to Scale Widget

**Ritorna**

Scale total tick count

`int32_t lv_scale_get_major_tick_every(lv_obj_t *obj)`

Get how often the major tick will be drawn

**Parametri**

- **obj** -- pointer to Scale Widget

**Ritorna**

Scale major tick every count

`int32_t lv_scale_get_rotation(lv_obj_t *obj)`

Get angular location of low end of Scale.

**Parametri**

- **obj** -- pointer to Scale Widget

**Ritorna**

Scale low end angular location

`bool lv_scale_get_label_show(lv_obj_t *obj)`

Gets label visibility

**Parametri**

- **obj** -- pointer to Scale Widget

**Ritorna**

true if tick label is enabled, false otherwise

`uint32_t lv_scale_get_angle_range(lv_obj_t *obj)`

Get Scale's range in degrees

**Parametri**

- **obj** -- pointer to Scale Widget

**Ritorna**

Scale's angle\_range

`int32_t lv_scale_get_range_min_value(lv_obj_t *obj)`

Get minimum value for Scale

**Parametri**

- **obj** -- pointer to Scale Widget

**Ritorna**

Scale's minimum value

`int32_t lv_scale_get_range_max_value(lv_obj_t *obj)`

Get maximum value for Scale

**Parametri**

- **obj** -- pointer to Scale Widget

**Ritorna**

Scale's maximum value

**Variables**const lv\_obj\_class\_t **lv\_scale\_class****lv\_scale\_private.h**struct **\_lv\_scale\_section\_t****Public Members**const *lv\_style\_t* \***main\_style**

Style to use for MAIN part(s) of scale when it falls within this section's range

const *lv\_style\_t* \***indicator\_style**

Style to use for INDICATOR part(s) of scale when it falls within this section's range

const *lv\_style\_t* \***items\_style**

Style to use for ITEMS part(s) of scale when it falls within this section's range

int32\_t **range\_min**

Scale parts with value &gt;= this value will be drawn using applicable style.

int32\_t **range\_max**

Scale parts with value &lt;= this value will be drawn using applicable style.

uint32\_t **first\_tick\_idx\_in\_section**

Internal (set during drawing): Tick index of first tick that falls within this section; LV\_SCALE\_TICK\_IDX\_DEFAULT\_ID if section contains no ticks.

uint32\_t **last\_tick\_idx\_in\_section**

Internal (set during drawing): Tick index of last tick that falls within this section; LV\_SCALE\_TICK\_IDX\_DEFAULT\_ID if section contains no ticks.

int32\_t **first\_tick\_in\_section\_width**

Internal (set during drawing)

int32\_t **last\_tick\_in\_section\_width**

Internal (set during drawing)

*lv\_point\_t* **first\_tick\_in\_section**

Internal (set during drawing)

*lv\_point\_t* **last\_tick\_in\_section**

Internal (set during drawing)

uint32\_t **first\_tick\_idx\_is\_major**Internal (set during drawing): true if **first\_tick\_idx\_in\_section** represents a major tick.

---

```
uint32_t last_tick_idx_is_major
```

Internal (set during drawing): true if `last_tick_idx_in_section` represents a major tick.

```
struct _lv_scale_t
```

#### Public Members

```
lv_obj_t obj
```

Base Widget part of Scale

```
lv_ll_t section_ll
```

Linked list for the sections (stores `lv_scale_section_t`)

```
const char **txt_src
```

Optional list of text strings for major ticks when custom labels are provided.

```
lv_scale_mode_t mode
```

Orientation and layout of scale.

```
int32_t range_min
```

Scale's minimum value

```
int32_t range_max
```

Scale's maximum value

```
uint32_t total_tick_count
```

Total number of ticks (major and minor)

```
uint32_t major_tick_every
```

Frequency of major ticks to minor ticks

```
uint32_t label_enabled
```

Draw labels for major ticks?

```
uint32_t post_draw
```

false: drawing occurs during `LV_EVENT_DRAW_MAIN`; true : drawing occurs during `LV_EVENT_DRAW_POST`.

```
uint32_t draw_ticks_on_top
```

Draw ticks on top of main line?

```
uint32_t angle_range
```

Degrees between low end and high end of scale

```
int32_t rotation
```

Clockwise angular offset from 3-o'clock position of low end of scale

```
int32_t custom_label_cnt
```

Number of custom labels provided in `txt_src`

`int32_t last_tick_width`

Width of last tick in pixels

`int32_t first_tick_width`

Width of first tick in pixels

## 14.24.27 slider

`lv_slider.h`

### Enums

enum `lv_slider_mode_t`

*Values:*

enumerator `LV_SLIDER_MODE_NORMAL`

enumerator `LV_SLIDER_MODE_SYMMETRICAL`

enumerator `LV_SLIDER_MODE_RANGE`

enum `lv_slider_orientation_t`

*Values:*

enumerator `LV_SLIDER_ORIENTATION_AUTO`

enumerator `LV_SLIDER_ORIENTATION_HORIZONTAL`

enumerator `LV_SLIDER_ORIENTATION_VERTICAL`

enum `_lv_property_slider_id_t`

*Values:*

enumerator `LV_PROPERTY_SLIDER_VALUE`

enumerator `LV_PROPERTY_SLIDER_LEFT_VALUE`

enumerator `LV_PROPERTY_SLIDER_RANGE`

enumerator `LV_PROPERTY_SLIDER_MIN_VALUE`

enumerator `LV_PROPERTY_SLIDER_MAX_VALUE`

enumerator `LV_PROPERTY_SLIDER_MODE`

enumerator `LV_PROPERTY_SLIDER_IS_DRAGGED`

enumerator `LV_PROPERTY_SLIDER_IS_SYMMETRICAL`

enumerator **LV\_PROPERTY\_SLIDER\_END**

## Functions

`lv_obj_t *lv_slider_create(lv_obj_t *parent)`

Create a slider object

### Parametri

**parent** -- pointer to an object, it will be the parent of the new slider.

### Ritorna

pointer to the created slider

`void lv_slider_set_value(lv_obj_t *obj, int32_t value, lv_anim_enable_t anim)`

Set a new value on the slider

### Parametri

- **obj** -- pointer to a slider object
- **value** -- the new value
- **anim** -- LV\_ANIM\_ON: set the value with an animation; LV\_ANIM\_OFF: change the value immediately

`void lv_slider_set_start_value(lv_obj_t *obj, int32_t value, lv_anim_enable_t anim)`

Set a new value for the left knob of a slider

### Parametri

- **obj** -- pointer to a slider object
- **value** -- new value
- **anim** -- LV\_ANIM\_ON: set the value with an animation; LV\_ANIM\_OFF: change the value immediately

`void lv_slider_set_range(lv_obj_t *obj, int32_t min, int32_t max)`

Set the minimum and the maximum values of a bar

### Parametri

- **obj** -- pointer to the slider object
- **min** -- minimum value
- **max** -- maximum value

`void lv_slider_set_min_value(lv_obj_t *obj, int32_t min)`

Set the minimum values of a bar

### Parametri

- **obj** -- pointer to the slider object
- **min** -- minimum value

`void lv_slider_set_max_value(lv_obj_t *obj, int32_t max)`

Set the maximum values of a bar

### Parametri

- **obj** -- pointer to the slider object
- **max** -- maximum value

`void lv_slider_set_mode(lv_obj_t *obj, lv_slider_mode_t mode)`

Set the mode of slider.

### Parametri

- **obj** -- pointer to a slider object
- **mode** -- the mode of the slider. See `lv_slider_mode_t`

`void lv_slider_set_orientation(lv_obj_t *obj, lv_slider_orientation_t orientation)`

Set the orientation of slider.

### Parametri

- **obj** -- pointer to a slider object
- **orientation** -- slider orientation from `lv_slider_orientation_t`

`int32_t lv_slider_get_value(const lv_obj_t *obj)`

Get the value of the main knob of a slider

**Parametri**

**obj** -- pointer to a slider object

**Ritorna**

the value of the main knob of the slider

`int32_t lv_slider_get_left_value(const lv_obj_t *obj)`

Get the value of the left knob of a slider

**Parametri**

**obj** -- pointer to a slider object

**Ritorna**

the value of the left knob of the slider

`int32_t lv_slider_get_min_value(const lv_obj_t *obj)`

Get the minimum value of a slider

**Parametri**

**obj** -- pointer to a slider object

**Ritorna**

the minimum value of the slider

`int32_t lv_slider_get_max_value(const lv_obj_t *obj)`

Get the maximum value of a slider

**Parametri**

**obj** -- pointer to a slider object

**Ritorna**

the maximum value of the slider

`bool lv_slider_is_dragged(const lv_obj_t *obj)`

Give the slider is being dragged or not

**Parametri**

**obj** -- pointer to a slider object

**Ritorna**

true: drag in progress false: not dragged

`lv_slider_mode_t lv_slider_get_mode(lv_obj_t *slider)`

Get the mode of the slider.

**Parametri**

**slider** -- pointer to a slider object

**Ritorna**

see `lv_slider_mode_t`

`lv_slider_orientation_t lv_slider_get_orientation(lv_obj_t *slider)`

Get the orientation of slider.

**Parametri**

**obj** -- pointer to a slider object

**Ritorna**

slider orientation from `lv_slider_orientation_t`

`bool lv_slider_is_symmetrical(lv_obj_t *obj)`

Give the slider is in symmetrical mode or not

**Parametri**

**obj** -- pointer to slider object

**Ritorna**

true: in symmetrical mode false : not in

## Variables

const lv\_obj\_class\_t **lv\_slider\_class**

**lv\_slider\_private.h**

struct **\_lv\_slider\_t**

### Public Members

lv\_bar\_t **bar**

Add the ancestor's type first

*lv\_area\_t* **left\_knob\_area**

*lv\_area\_t* **right\_knob\_area**

*lv\_point\_t* **pressed\_point**

int32\_t \***value\_to\_set**

Which bar value to set

uint8\_t **dragging**

1: the slider is being dragged

uint8\_t **left\_knob\_focus**

1: with encoder now the right knob can be adjusted

## 14.24.28 span

**lv\_span.h**

### Typedef

typedef struct *lv\_span\_coords\_t* **lv\_span\_coords\_t**

Coords of a span

### Enums

enum **lv\_span\_overflow\_t**

Values:

enumerator **LV\_SPAN\_OVERFLOW\_CLIP**

enumerator **LV\_SPAN\_OVERFLOW\_ELLIPSIS**

enumerator **LV\_SPAN\_OVERFLOW\_LAST**

Fence member

---

enum **lv\_span\_mode\_t**

*Values:*

enumerator **LV\_SPAN\_MODE\_FIXED**

fixed the obj size

enumerator **LV\_SPAN\_MODE\_EXPAND**

Expand the object size to the text size

enumerator **LV\_SPAN\_MODE\_BREAK**

Keep width, break the too long lines and expand height

enumerator **LV\_SPAN\_MODE\_LAST**

Fence member

## Functions

void **lv\_span\_stack\_init**(void)

void **lv\_span\_stack\_deinit**(void)

lv\_obj\_t \***lv\_spangroup\_create**(lv\_obj\_t \*parent)

Create a spangroup object

**Parametri**

- **parent** -- pointer to an object, it will be the parent of the new spangroup

**Ritorna**

- pointer to the created spangroup

lv\_span\_t \***lv\_spangroup\_add\_span**(lv\_obj\_t \*obj)

Create a span string descriptor and add to spangroup.

**Parametri**

- **obj** -- pointer to a spangroup object.

**Ritorna**

- pointer to the created span.

void **lv\_spangroup\_delete\_span**(lv\_obj\_t \*obj, lv\_span\_t \*span)

Remove the span from the spangroup and free memory.

**Parametri**

- **obj** -- pointer to a spangroup object.
- **span** -- pointer to a span.

void **lv\_span\_set\_text**(lv\_span\_t \*span, const char \*text)

Set a new text for a span. Memory will be allocated to store the text by the span. As the spangroup is not passed a redraw (invalidation) can't be triggered automatically. Therefore **lv\_spangroup\_refresh(spangroup)** needs to be called manually,

**Parametri**

- **span** -- pointer to a span.
- **text** -- pointer to a text.

void **lv\_span\_set\_text\_static**(lv\_span\_t \*span, const char \*text)

Set a static text. It will not be saved by the span so the 'text' variable has to be 'alive' while the span exist. As the spangroup is not passed a redraw (invalidation) can't be triggered automatically. Therefore **lv\_spangroup\_refresh(spangroup)** needs to be called manually,

Set a static text. It will not be saved by the span so the 'text' variable has to be 'alive' while the span exist.

**Parametri**

- **span** -- pointer to a span.

- **text** -- pointer to a text.
- **span** -- pointer to a span.
- **text** -- pointer to a text.

**void lv\_spangroup\_set\_span\_text(lv\_obj\_t \*obj, lv\_span\_t \*span, const char \*text)**

Set a new text for a span. Memory will be allocated to store the text by the span.

**Parametri**

- **obj** -- pointer to a spangroup widget.
- **span** -- pointer to a span.
- **text** -- pointer to a text.

**void lv\_spangroup\_set\_span\_text\_static(lv\_obj\_t \*obj, lv\_span\_t \*span, const char \*text)**

Set a new text for a span. Memory will be allocated to store the text by the span.

**Parametri**

- **obj** -- pointer to a spangroup widget.
- **span** -- pointer to a span.
- **text** -- pointer to a text.

**void lv\_spangroup\_set\_span\_style(lv\_obj\_t \*obj, lv\_span\_t \*span, const lv\_style\_t \*style)**

Copy all style properties of style to the bbuilt-in static style of the span.

**Parametri**

- **obj** -- pointer\_to a spangroup
- **span** -- pointer to a span.
- **style** -- pointer to a style to copy into the span's built-in style

**void lv\_spangroup\_set\_align(lv\_obj\_t \*obj, lv\_text\_align\_t align)**

DEPRECATED. Use the text\_align style property instead Set the align of the spangroup.

**Parametri**

- **obj** -- pointer to a spangroup object.
- **align** -- see [lv\\_text\\_align\\_t](#) for details.

**void lv\_spangroup\_set\_overflow(lv\_obj\_t \*obj, lv\_span\_overflow\_t overflow)**

Set the overflow of the spangroup.

**Parametri**

- **obj** -- pointer to a spangroup object.
- **overflow** -- see [lv\\_span\\_overflow\\_t](#) for details.

**void lv\_spangroup\_set\_indent(lv\_obj\_t \*obj, int32\_t indent)**

Set the indent of the spangroup.

**Parametri**

- **obj** -- pointer to a spangroup object.
- **indent** -- the first line indentation

**void lv\_spangroup\_set\_mode(lv\_obj\_t \*obj, lv\_span\_mode\_t mode)**

DEPRECATED, set the width to LV\_SIZE\_CONTENT or fixed value to control expanding/wrapping" Set the mode of the spangroup.

**Parametri**

- **obj** -- pointer to a spangroup object.
- **mode** -- see [lv\\_span\\_mode\\_t](#) for details.

**void lv\_spangroup\_set\_max\_lines(lv\_obj\_t \*obj, int32\_t lines)**

Set maximum lines of the spangroup.

**Parametri**

- **obj** -- pointer to a spangroup object.
- **lines** -- max lines that can be displayed in LV\_SPAN\_MODE\_BREAK mode. < 0 means no limit.

**lv\_style\_t \*lv\_span\_get\_style(lv\_span\_t \*span)**

Get a pointer to the style of a span's built-in style. Any lv\_style\_set\_... functions can be applied on the returned style.

**Parametri**  
**span** -- pointer to the span

**Ritorna**  
pointer to the style. (valid as long as the span is valid)

```
const char *lv_span_get_text(lv_span_t *span)
```

Get a pointer to the text of a span

**Parametri**  
**span** -- pointer to the span

**Ritorna**  
pointer to the text

```
lv_span_t *lv_spangroup_get_child(const lv_obj_t *obj, int32_t id)
```

Get a spangroup child by its index.

**Parametri**

- **obj** -- The spangroup object
- **id** -- the index of the child. 0: the oldest (firstly created) child 1: the second oldest child count-1: the youngest -1: the youngest -2: the second youngest

**Ritorna**  
The child span at index **id**, or NULL if the ID does not exist

```
uint32_t lv_spangroup_get_span_count(const lv_obj_t *obj)
```

Get number of spans

**Parametri**  
**obj** -- the spangroup object to get the child count of.

**Ritorna**  
the span count of the spangroup.

```
lv_text_align_t lv_spangroup_get_align(lv_obj_t *obj)
```

Get the align of the spangroup.

**Parametri**  
**obj** -- pointer to a spangroup object.

**Ritorna**  
the align value.

```
lv_span_overflow_t lv_spangroup_get_overflow(lv_obj_t *obj)
```

Get the overflow of the spangroup.

**Parametri**  
**obj** -- pointer to a spangroup object.

**Ritorna**  
the overflow value.

```
int32_t lv_spangroup_get_indent(lv_obj_t *obj)
```

Get the indent of the spangroup.

**Parametri**  
**obj** -- pointer to a spangroup object.

**Ritorna**  
the indent value.

```
lv_span_mode_t lv_spangroup_get_mode(lv_obj_t *obj)
```

Get the mode of the spangroup.

**Parametri**  
**obj** -- pointer to a spangroup object.

```
int32_t lv_spangroup_get_max_lines(lv_obj_t *obj)
```

Get maximum lines of the spangroup.

**Parametri**  
**obj** -- pointer to a spangroup object.

**Ritorna**  
the max lines value.

`int32_t lv_spangroup_get_max_line_height(lv_obj_t *obj)`

Get max line height of all span in the spangroup.

**Parametri**

**obj** -- pointer to a spangroup object.

`uint32_t lv_spangroup_get_expand_width(lv_obj_t *obj, uint32_t max_width)`

Get the text content width when all span of spangroup on a line.

**Parametri**

- **obj** -- pointer to a spangroup object.
- **max\_width** -- if text content width  $\geq$  max\_width, return max\_width to reduce computation, if max\_width == 0, returns the text content width.

**Ritorna**

text content width or max\_width.

`int32_t lv_spangroup_get_expand_height(lv_obj_t *obj, int32_t width)`

Get the text content height with width fixed.

**Parametri**

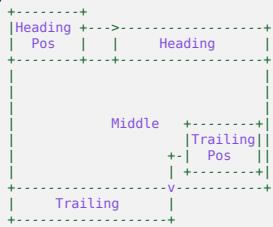
- **obj** -- pointer to a spangroup object.
- **width** -- the width of the span group.

`lv_span_coords_t lv_spangroup_get_span_coords(lv_obj_t *obj, const lv_span_t *span)`

Get the span's coords in the spangroup.

**Nota**

Before calling this function, please make sure that the layout of span group has been updated. Like calling `lv_obj_update_layout()` like function.



**Parametri**

- **obj** -- pointer to a spangroup object.
- **span** -- pointer to a span.

**Ritorna**

the span's coords in the spangroup.

`lv_span_t *lv_spangroup_get_span_by_point(lv_obj_t *obj, const lv_point_t *point)`

Get the span object by point.

**Parametri**

- **obj** -- pointer to a spangroup object.
- **point** -- pointer to point containing absolute coordinates

**Ritorna**

pointer to the span under the point or **NULL** if not found.

`void lv_spangroup_refresh(lv_obj_t *obj)`

Update the mode of the spangroup.

**Parametri**

**obj** -- pointer to a spangroup object.

## Variables

const lv\_obj\_class\_t **lv\_spangroup\_class**

struct **\_lv\_span\_coords\_t**

#include <lv\_span.h> Coords of a span

### Public Members

*lv\_area\_t* **heading**

*lv\_area\_t* **middle**

*lv\_area\_t* **trailing**

## lv\_span\_private.h

struct **\_lv\_span\_t**

### Public Members

char \***txt**

a pointer to display text

*lv\_style\_t* **style**

display text style

uint32\_t **static\_flag**

the text is static flag

*lv\_point\_t* **trailing\_pos**

int32\_t **trailing\_height**

struct **\_lv\_spangroup\_t**

#include <lv\_span\_private.h> Data of label

### Public Members

lv\_obj\_t **obj**

int32\_t **lines**

int32\_t **indent**

first line indent

int32\_t **cache\_w**

the cache automatically calculates the width

```

int32_t cache_h
    similar cache_w

lv_ll_t child_ll

uint32_t overflow
    details see lv_span_overflow_t

uint32_t refresh
    the spangroup need refresh cache_w and cache_h

```

## 14.24.29 spinbox

**lv\_spinbox.h**

### Defines

**LV\_SPINBOX\_MAX\_DIGIT\_COUNT**

### Functions

**lv\_obj\_t \*lv\_spinbox\_create(lv\_obj\_t \*parent)**

Create a spinbox object

#### Parametri

- **parent** -- pointer to an object, it will be the parent of the new spinbox

#### Ritorna

pointer to the created spinbox

**void lv\_spinbox\_set\_value(lv\_obj\_t \*obj, int32\_t v)**

Set spinbox value

#### Parametri

- **obj** -- pointer to spinbox
- **v** -- value to be set

**void lv\_spinbox\_set\_rollover(lv\_obj\_t \*obj, bool rollover)**

Set spinbox rollover function

#### Parametri

- **obj** -- pointer to spinbox
- **rollover** -- true or false to enable or disable (default)

**void lv\_spinbox\_set\_digit\_format(lv\_obj\_t \*obj, uint32\_t digit\_count, uint32\_t sep\_pos)**

Set spinbox digit format (digit count and decimal format)

#### Parametri

- **obj** -- pointer to spinbox
- **digit\_count** -- number of digit excluding the decimal separator and the sign
- **sep\_pos** -- number of digit before the decimal point. If 0, decimal point is not shown

**void lv\_spinbox\_set\_step(lv\_obj\_t \*obj, uint32\_t step)**

Set spinbox step

#### Parametri

- **obj** -- pointer to spinbox
- **step** -- steps on increment/decrement. Can be 1, 10, 100, 1000, etc the digit that will change.

**void lv\_spinbox\_set\_range(lv\_obj\_t \*obj, int32\_t range\_min, int32\_t range\_max)**

Set spinbox value range

**Parametri**

- **obj** -- pointer to spinbox
- **range\_min** -- maximum value, inclusive
- **range\_max** -- minimum value, inclusive

**void lv\_spinbox\_set\_cursor\_pos(lv\_obj\_t \*obj, uint32\_t pos)**

Set cursor position to a specific digit for edition

**Parametri**

- **obj** -- pointer to spinbox
- **pos** -- selected position in spinbox

**void lv\_spinbox\_set\_digit\_step\_direction(lv\_obj\_t \*obj, lv\_dir\_t direction)**

Set direction of digit step when clicking an encoder button while in editing mode

**Parametri**

- **obj** -- pointer to spinbox
- **direction** -- the direction (LV\_DIR\_RIGHT or LV\_DIR\_LEFT)

**bool lv\_spinbox\_get\_rollover(lv\_obj\_t \*obj)**

Get spinbox rollover function status

**Parametri**

- **obj** -- pointer to spinbox

**int32\_t lv\_spinbox\_get\_value(lv\_obj\_t \*obj)**

Get the spinbox numeral value (user has to convert to float according to its digit format)

**Parametri**

- **obj** -- pointer to spinbox

**Ritorna**

value integer value of the spinbox

**int32\_t lv\_spinbox\_get\_step(lv\_obj\_t \*obj)**

Get the spinbox step value (user has to convert to float according to its digit format)

**Parametri**

- **obj** -- pointer to spinbox

**Ritorna**

value integer step value of the spinbox

**void lv\_spinbox\_step\_next(lv\_obj\_t \*obj)**

Select next lower digit for edition by dividing the step by 10

**Parametri**

- **obj** -- pointer to spinbox

**void lv\_spinbox\_step\_prev(lv\_obj\_t \*obj)**

Select next higher digit for edition by multiplying the step by 10

**Parametri**

- **obj** -- pointer to spinbox

**void lv\_spinbox\_increment(lv\_obj\_t \*obj)**

Increment spinbox value by one step

**Parametri**

- **obj** -- pointer to spinbox

**void lv\_spinbox\_decrement(lv\_obj\_t \*obj)**

Decrement spinbox value by one step

**Parametri**

- **obj** -- pointer to spinbox

## Variables

const lv\_obj\_class\_t **lv\_spinbox\_class**

### lv\_spinbox\_private.h

struct **\_lv\_spinbox\_t**

#include <lv\_spinbox\_private.h> Data of spinbox

#### Public Members

lv\_textarea\_t **ta**

Ext. of ancestor

int32\_t **value**

int32\_t **range\_max**

int32\_t **range\_min**

int32\_t **step**

uint32\_t **digit\_count**

uint32\_t **dec\_point\_pos**

if 0, there is no separator and the number is an integer

uint32\_t **rollover**

Set to true for rollover functionality

uint32\_t **digit\_step\_dir**

the direction the digit will step on encoder button press when editing

## 14.24.30 spinner

### lv\_spinner.h

#### Functions

lv\_obj\_t \***lv\_spinner\_create**(lv\_obj\_t \*parent)

Create a spinner widget

##### Parametri

**parent** -- pointer to an object, it will be the parent of the new spinner.

##### Ritorna

the created spinner

void **lv\_spinner\_set\_anim\_params**(lv\_obj\_t \*obj, uint32\_t t, uint32\_t angle)

Set the animation time and arc length of the spinner

##### Parametri

- **obj** -- pointer to a spinner
- **t** -- the animation time in milliseconds
- **angle** -- the angle of the arc in degrees

## Variables

```
const lv_obj_class_t lv_spinner_class
```

### 14.24.31 switch

**lv\_switch.h**

#### Defines

**LV\_SWITCH\_KNOB\_EXT\_AREA\_CORRECTION**

Switch knob extra area correction factor

#### Enums

enum **lv\_switch\_orientation\_t**

*Values:*

enumerator **LV\_SWITCH\_ORIENTATION\_AUTO**

enumerator **LV\_SWITCH\_ORIENTATION\_HORIZONTAL**

enumerator **LV\_SWITCH\_ORIENTATION\_VERTICAL**

#### Functions

**lv\_obj\_t \*lv\_switch\_create(lv\_obj\_t \*parent)**

Create a switch object

**Parametri**

**parent** -- pointer to an object, it will be the parent of the new switch

**Ritorna**

pointer to the created switch

**void lv\_switch\_set\_orientation(lv\_obj\_t \*obj, lv\_switch\_orientation\_t orientation)**

Set the orientation of switch.

**Parametri**

- **obj** -- pointer to switch object

- **orientation** -- switch orientation from `lv_switch_orientation_t`

*lv\_switch\_orientation\_t* **lv\_switch\_get\_orientation(lv\_obj\_t \*obj)**

Get the orientation of switch.

**Parametri**

**obj** -- pointer to switch object

**Ritorna**

switch orientation from `::lv_switch_orientation_t`

## Variables

```
const lv_obj_class_t lv_switch_class
```

**lv\_switch\_private.h**

```
struct _lv_switch_t
```

**Public Members**

`lv_obj_t obj`

`int32_t anim_state`

`lv_switch_orientation_t orientation`

Orientation of switch

**14.24.32 table**

`lv_table.h`

**Defines**

`LV_TABLE_CELL_NONE`

**Enums**

enum `lv_table_cell_ctrl_t`

*Values:*

enumerator `LV_TABLE_CELL_CTRL_NONE`

enumerator `LV_TABLE_CELL_CTRL_MERGE_RIGHT`

enumerator `LV_TABLE_CELL_CTRL_TEXT_CROP`

enumerator `LV_TABLE_CELL_CTRL_CUSTOM_1`

enumerator `LV_TABLE_CELL_CTRL_CUSTOM_2`

enumerator `LV_TABLE_CELL_CTRL_CUSTOM_3`

enumerator `LV_TABLE_CELL_CTRL_CUSTOM_4`

**Functions**

`lv_obj_t *lv_table_create(lv_obj_t *parent)`

Create a table object

**Parametri**

`parent` -- pointer to an object, it will be the parent of the new table

**Ritorna**

pointer to the created table

`void lv_table_set_cell_value(lv_obj_t *obj, uint32_t row, uint32_t col, const char *txt)`

Set the value of a cell.

 **Nota**

New rows/columns are added automatically if required

**Parametri**

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row\_cnt -1]
- **col** -- id of the column [0 .. col\_cnt -1]
- **txt** -- text to display in the cell. It will be copied and saved so this variable is not required after this function call.

**void lv\_table\_set\_cell\_value\_fmt(lv\_obj\_t \*obj, uint32\_t row, uint32\_t col, const char \*fmt, ...)**

Set the value of a cell. Memory will be allocated to store the text by the table.

 **Nota**

New rows/columns are added automatically if required

**Parametri**

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row\_cnt -1]
- **col** -- id of the column [0 .. col\_cnt -1]
- **fmt** -- printf-like format

**void lv\_table\_set\_row\_count(lv\_obj\_t \*obj, uint32\_t row\_cnt)**

Set the number of rows

**Parametri**

- **obj** -- table pointer to a Table object
- **row\_cnt** -- number of rows

**void lv\_table\_set\_column\_count(lv\_obj\_t \*obj, uint32\_t col\_cnt)**

Set the number of columns

**Parametri**

- **obj** -- table pointer to a Table object
- **col\_cnt** -- number of columns.

**void lv\_table\_set\_column\_width(lv\_obj\_t \*obj, uint32\_t col\_id, int32\_t w)**

Set the width of a column

**Parametri**

- **obj** -- table pointer to a Table object
- **col\_id** -- id of the column [0 .. LV\_TABLE\_COL\_MAX -1]
- **w** -- width of the column

**void lv\_table\_set\_cell\_ctrl(lv\_obj\_t \*obj, uint32\_t row, uint32\_t col, lv\_table\_cell\_ctrl\_t ctrl)**

Add control bits to the cell.

**Parametri**

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row\_cnt -1]
- **col** -- id of the column [0 .. col\_cnt -1]
- **ctrl** -- OR-ed values from ::lv\_table\_cell\_ctrl\_t

**void lv\_table\_clear\_cell\_ctrl(lv\_obj\_t \*obj, uint32\_t row, uint32\_t col, lv\_table\_cell\_ctrl\_t ctrl)**

Clear control bits of the cell.

**Parametri**

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row\_cnt -1]
- **col** -- id of the column [0 .. col\_cnt -1]
- **ctrl** -- OR-ed values from ::lv\_table\_cell\_ctrl\_t

**void lv\_table\_set\_cell\_user\_data(lv\_obj\_t \*obj, uint16\_t row, uint16\_t col, void \*user\_data)**

Add custom user data to the cell.

**Parametri**

- **obj** -- pointer to a Table object

- **row** -- id of the row [0 .. row\_cnt -1]
- **col** -- id of the column [0 .. col\_cnt -1]
- **user\_data** -- pointer to the new user\_data. Should be allocated by `lv_malloc`, and it will be freed automatically when the table is deleted or when the cell is dropped due to lower row or column count.

`void lv_table_set_selected_cell(lv_obj_t *obj, uint16_t row, uint16_t col)`

Set the selected cell

**Parametri**

- **obj** -- pointer to a table object
- **row** -- id of the cell row to select
- **col** -- id of the cell column to select

`const char *lv_table_get_cell_value(lv_obj_t *obj, uint32_t row, uint32_t col)`

Get the value of a cell.

**Parametri**

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row\_cnt -1]
- **col** -- id of the column [0 .. col\_cnt -1]

**Ritorna**

text in the cell

`uint32_t lv_table_get_row_count(lv_obj_t *obj)`

Get the number of rows.

**Parametri**

**obj** -- table pointer to a Table object

**Ritorna**

number of rows.

`uint32_t lv_table_get_column_count(lv_obj_t *obj)`

Get the number of columns.

**Parametri**

**obj** -- table pointer to a Table object

**Ritorna**

number of columns.

`int32_t lv_table_get_column_width(lv_obj_t *obj, uint32_t col)`

Get the width of a column

**Parametri**

- **obj** -- table pointer to a Table object
- **col** -- id of the column [0 .. LV\_TABLE\_COL\_MAX -1]

**Ritorna**

width of the column

`bool lv_table_has_cell_ctrl(lv_obj_t *obj, uint32_t row, uint32_t col, lv_table_cell_ctrl_t ctrl)`

Get whether a cell has the control bits

**Parametri**

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row\_cnt -1]
- **col** -- id of the column [0 .. col\_cnt -1]
- **ctrl** -- OR-ed values from ::lv\_table\_cell\_ctrl\_t

**Ritorna**

true: all control bits are set; false: not all control bits are set

`void lv_table_get_selected_cell(lv_obj_t *obj, uint32_t *row, uint32_t *col)`

Get the selected cell (pressed and or focused)

**Parametri**

- **obj** -- pointer to a table object
- **row** -- pointer to variable to store the selected row (LV\_TABLE\_CELL\_NONE: if no cell selected)

- **col** -- pointer to variable to store the selected column (LV\_TABLE\_CELL\_NONE: if no cell selected)

```
void *lv_table_get_cell_user_data(lv_obj_t *obj, uint16_t row, uint16_t col)
```

Get custom user data to the cell.

#### Parametri

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row\_cnt -1]
- **col** -- id of the column [0 .. col\_cnt -1]

## Variables

```
const lv_obj_class_t lv_table_class
```

## lv\_table\_private.h

```
struct _lv_table_cell_t
#include <lv_table_private.h> Cell data
```

#### Public Members

*lv\_table\_cell\_ctrl\_t* **ctrl**

void \***user\_data**

Custom user data

char **txt**[1]

Variable length array

```
struct _lv_table_t
```

#include <lv\_table\_private.h> Table data

#### Public Members

lv\_obj\_t **obj**

uint32\_t **col\_cnt**

uint32\_t **row\_cnt**

lv\_table\_cell\_t \*\***cell\_data**

int32\_t \***row\_h**

int32\_t \***col\_w**

uint32\_t **col\_act**

uint32\_t **row\_act**

### 14.24.33 tabview

`lv_tabview.h`

#### Functions

`lv_obj_t *lv_tabview_create(lv_obj_t *parent)`

Create a tabview widget

##### Parametri

- **parent** -- pointer to a parent widget

##### Ritorna

the created tabview

`lv_obj_t *lv_tabview_add_tab(lv_obj_t *obj, const char *name)`

Add a tab to the tabview

##### Parametri

- **obj** -- pointer to a tabview widget

- **name** -- the name of the tab, it will be displayed on the tab bar

##### Ritorna

the widget where the content of the tab can be created

`void lv_tabview_rename_tab(lv_obj_t *obj, uint32_t idx, const char *new_name)`

Change the name of the tab

##### Parametri

- **obj** -- pointer to a tabview widget

- **idx** -- the index of the tab to rename

- **new\_name** -- the new name as a string

`void lv_tabview_set_active(lv_obj_t *obj, uint32_t idx, lv_anim_enable_t anim_en)`

Show a tab

##### Parametri

- **obj** -- pointer to a tabview widget

- **idx** -- the index of the tab to show

- **anim\_en** -- LV\_ANIM\_ON/OFF

`void lv_tabview_set_tab_bar_position(lv_obj_t *obj, lv_dir_t dir)`

Set the position of the tab bar

##### Parametri

- **obj** -- pointer to a tabview widget

- **dir** -- LV\_DIR\_TOP/BOTTOM/LEFT/RIGHT

`void lv_tabview_set_tab_bar_size(lv_obj_t *obj, int32_t size)`

Set the width or height of the tab bar

##### Parametri

- **obj** -- pointer to tabview widget

- **size** -- size of the tab bar in pixels or percentage. will be used as width or height based on the position of the tab bar)

`uint32_t lv_tabview_get_tab_count(lv_obj_t *obj)`

Get the number of tabs

##### Parametri

- **obj** -- pointer to a tabview widget

##### Ritorna

the number of tabs

`uint32_t lv_tabview_get_tab_active(lv_obj_t *obj)`

Get the current tab's index

##### Parametri

- **obj** -- pointer to a tabview widget

##### Ritorna

the zero based index of the current tab

`lv_obj_t *lv_tabview_get_content(lv_obj_t *obj)`

Get the widget where the container of each tab is created

**Parametri**

**obj** -- pointer to a tabview widget

**Ritorna**

the main container widget

`lv_obj_t *lv_tabview_get_tab_bar(lv_obj_t *obj)`

Get the tab bar where the buttons are created

**Parametri**

**obj** -- pointer to a tabview widget

**Ritorna**

the tab bar

## Variables

const lv\_obj\_class\_t **lv\_tabview\_class**

**lv\_tabview\_private.h**

struct **\_lv\_tabview\_t**

### Public Members

`lv_obj_t obj`

`uint32_t tab_cur`

`lv_dir_t tab_pos`

## 14.24.34 textarea

**lv\_textarea.h**

### Defines

**LV\_TEXTAREA\_CURSOR\_LAST**

**LV\_PART\_TEXTAREA\_PLACEHOLDER**

### Enums

enum **\_lv\_property\_textarea\_id\_t**

*Values:*

enumerator **LV\_PROPERTY\_TEXTAREA\_TEXT**

enumerator **LV\_PROPERTY\_TEXTAREA\_PLACEHOLDER\_TEXT**

enumerator **LV\_PROPERTY\_TEXTAREA\_CURSOR\_POS**

enumerator **LV\_PROPERTY\_TEXTAREA\_CURSOR\_CLICK\_POS**

enumerator **LV\_PROPERTY\_TEXTAREA\_PASSWORD\_MODE**

enumerator **LV\_PROPERTY\_TEXTAREA\_PASSWORD\_BULLET**

enumerator **LV\_PROPERTY\_TEXTAREA\_ONE\_LINE**

enumerator **LV\_PROPERTY\_TEXTAREA\_ACCEPTED\_CHARS**

enumerator **LV\_PROPERTY\_TEXTAREA\_MAX\_LENGTH**

enumerator **LV\_PROPERTY\_TEXTAREA\_INSERT\_REPLACE**

enumerator **LV\_PROPERTY\_TEXTAREA\_TEXT\_SELECTION**

enumerator **LV\_PROPERTY\_TEXTAREA\_PASSWORD\_SHOW\_TIME**

enumerator **LV\_PROPERTY\_TEXTAREA\_LABEL**

enumerator **LV\_PROPERTY\_TEXTAREA\_TEXT\_IS\_SELECTED**

enumerator **LV\_PROPERTY\_TEXTAREA\_CURRENT\_CHAR**

enumerator **LV\_PROPERTY\_TEXTAREA\_END**

## Functions

`lv_obj_t *lv_textarea_create(lv_obj_t *parent)`

Create a text area object

**Parametri**

• **parent** -- pointer to an object, it will be the parent of the new text area

**Ritorna**

pointer to the created text area

`void lv_textarea_add_char(lv_obj_t *obj, uint32_t c)`

Insert a character to the current cursor position. To add a wide char, e.g. 'A' use `lv_text_encoded_conv_wc('A')`

**Parametri**

- **obj** -- pointer to a text area object
- **c** -- a character (e.g. 'a')

`void lv_textarea_add_text(lv_obj_t *obj, const char *txt)`

Insert a text to the current cursor position

**Parametri**

- **obj** -- pointer to a text area object
- **txt** -- a '\0' terminated string to insert

`void lv_textarea_delete_char(lv_obj_t *obj)`

Delete a the left character from the current cursor position

**Parametri**

• **obj** -- pointer to a text area object

**void lv\_textarea\_delete\_char\_forward(lv\_obj\_t \*obj)**

Delete the right character from the current cursor position

**Parametri**

- **obj** -- pointer to a text area object

**void lv\_textarea\_set\_text(lv\_obj\_t \*obj, const char \*txt)**

Set the text of a text area

**Parametri**

- **obj** -- pointer to a text area object
- **txt** -- pointer to the text

**void lv\_textarea\_set\_placeholder\_text(lv\_obj\_t \*obj, const char \*txt)**

Set the placeholder text of a text area

**Parametri**

- **obj** -- pointer to a text area object
- **txt** -- pointer to the text

**void lv\_textarea\_set\_cursor\_pos(lv\_obj\_t \*obj, int32\_t pos)**

Set the cursor position

**Parametri**

- **obj** -- pointer to a text area object
- **pos** -- the new cursor position in character index < 0 : index from the end of the text  
LV\_TEXTAREA\_CURSOR\_LAST: go after the last character

**void lv\_textarea\_set\_cursor\_click\_pos(lv\_obj\_t \*obj, bool en)**

Enable/Disable the positioning of the cursor by clicking the text on the text area.

**Parametri**

- **obj** -- pointer to a text area object
- **en** -- true: enable click positions; false: disable

**void lv\_textarea\_set\_password\_mode(lv\_obj\_t \*obj, bool en)**

Enable/Disable password mode

**Parametri**

- **obj** -- pointer to a text area object
- **en** -- true: enable, false: disable

**void lv\_textarea\_set\_password\_bullet(lv\_obj\_t \*obj, const char \*bullet)**

Set the replacement characters to show in password mode

**Parametri**

- **obj** -- pointer to a text area object
- **bullet** -- pointer to the replacement text

**void lv\_textarea\_set\_one\_line(lv\_obj\_t \*obj, bool en)**

Configure the text area to one line or back to normal

**Parametri**

- **obj** -- pointer to a text area object
- **en** -- true: one line, false: normal

**void lv\_textarea\_set\_accepted\_chars(lv\_obj\_t \*obj, const char \*list)**

Set a list of characters. Only these characters will be accepted by the text area

**Parametri**

- **obj** -- pointer to a text area object
- **list** -- list of characters. Only the pointer is saved. E.g. "+-,0123456789"

**void lv\_textarea\_set\_max\_length(lv\_obj\_t \*obj, uint32\_t num)**

Set max length of a Text Area.

**Parametri**

- **obj** -- pointer to a text area object
- **num** -- the maximal number of characters can be added  
(*lv\_textarea\_set\_text* ignores it)

**void lv\_textarea\_set\_insert\_replace(lv\_obj\_t \*obj, const char \*txt)**

In LV\_EVENT\_INSERT the text which planned to be inserted can be replaced by another text. It can be used to add automatic formatting to the text area.

**Parametri**

- **obj** -- pointer to a text area object
- **txt** -- pointer to a new string to insert. If "" no text will be added. The variable must be live after the `event_cb` exists. (Should be `global` or `static`)

**void lv\_textarea\_set\_text\_selection(lv\_obj\_t \*obj, bool en)**

Enable/disable selection mode.

**Parametri**

- **obj** -- pointer to a text area object
- **en** -- true or false to enable/disable selection mode

**void lv\_textarea\_set\_password\_show\_time(lv\_obj\_t \*obj, uint32\_t time)**

Set how long show the password before changing it to '\*'.

**Parametri**

- **obj** -- pointer to a text area object
- **time** -- show time in milliseconds. 0: hide immediately.

**void lv\_textarea\_set\_align(lv\_obj\_t \*obj, *lv\_text\_align\_t* align)**

*Deprecated:*

Use the normal `text_align` style property instead Set the label's alignment. It sets where the label is aligned (in one line mode it can be smaller than the text area) and how the lines of the area align in case of multiline text area

**Parametri**

- **obj** -- pointer to a text area object
- **align** -- the align mode from `lv_text_align_t`

**const char \*lv\_textarea\_get\_text(const lv\_obj\_t \*obj)**

Get the text of a text area. In password mode it gives the real text (not '\*'s).

**Parametri**

**obj** -- pointer to a text area object

**Ritorna**

pointer to the text

**const char \*lv\_textarea\_get\_placeholder\_text(lv\_obj\_t \*obj)**

Get the placeholder text of a text area

**Parametri**

**obj** -- pointer to a text area object

**Ritorna**

pointer to the text

**lv\_obj\_t \*lv\_textarea\_get\_label(const lv\_obj\_t \*obj)**

Get the label of a text area

**Parametri**

**obj** -- pointer to a text area object

**Ritorna**

pointer to the label object

**uint32\_t lv\_textarea\_get\_cursor\_pos(const lv\_obj\_t \*obj)**

Get the current cursor position in character index

**Parametri**

**obj** -- pointer to a text area object

**Ritorna**

the cursor position

**bool lv\_textarea\_get\_cursor\_click\_pos(lv\_obj\_t \*obj)**

Get whether the cursor click positioning is enabled or not.

**Parametri**  
**obj** -- pointer to a text area object

**Ritorna**  
true: enable click positions; false: disable

bool **lv\_textarea\_get\_password\_mode**(const lv\_obj\_t \*obj)

Get the password mode attribute

**Parametri**  
**obj** -- pointer to a text area object

**Ritorna**  
true: password mode is enabled, false: disabled

const char \***lv\_textarea\_get\_password\_bullet**(lv\_obj\_t \*obj)

Get the replacement characters to show in password mode

**Parametri**  
**obj** -- pointer to a text area object

**Ritorna**  
pointer to the replacement text

bool **lv\_textarea\_get\_one\_line**(const lv\_obj\_t \*obj)

Get the one line configuration attribute

**Parametri**  
**obj** -- pointer to a text area object

**Ritorna**  
true: one line configuration is enabled, false: disabled

const char \***lv\_textarea\_get\_accepted\_chars**(lv\_obj\_t \*obj)

Get a list of accepted characters.

**Parametri**  
**obj** -- pointer to a text area object

**Ritorna**  
list of accented characters.

uint32\_t **lv\_textarea\_get\_max\_length**(lv\_obj\_t \*obj)

Get max length of a Text Area.

**Parametri**  
**obj** -- pointer to a text area object

**Ritorna**  
the maximal number of characters to be add

bool **lv\_textarea\_text\_is\_selected**(const lv\_obj\_t \*obj)

Find whether text is selected or not.

**Parametri**  
**obj** -- pointer to a text area object

**Ritorna**  
whether text is selected or not

bool **lv\_textarea\_get\_text\_selection**(lv\_obj\_t \*obj)

Find whether selection mode is enabled.

**Parametri**  
**obj** -- pointer to a text area object

**Ritorna**  
true: selection mode is enabled, false: disabled

uint32\_t **lv\_textarea\_get\_password\_show\_time**(lv\_obj\_t \*obj)

Set how long show the password before changing it to '\*'

**Parametri**  
**obj** -- pointer to a text area object

**Ritorna**  
show time in milliseconds. 0: hide immediately.

`uint32_t lv_textarea_get_current_char(lv_obj_t *obj)`

Get a the character from the current cursor position

**Parametri**

**obj** -- pointer to a text area object

**Ritorna**

a the character or 0

`void lv_textarea_clear_selection(lv_obj_t *obj)`

Clear the selection on the text area.

**Parametri**

**obj** -- pointer to a text area object

`void lv_textarea_cursor_right(lv_obj_t *obj)`

Move the cursor one character right

**Parametri**

**obj** -- pointer to a text area object

`void lv_textarea_cursor_left(lv_obj_t *obj)`

Move the cursor one character left

**Parametri**

**obj** -- pointer to a text area object

`void lv_textarea_cursor_down(lv_obj_t *obj)`

Move the cursor one line down

**Parametri**

**obj** -- pointer to a text area object

`void lv_textarea_cursor_up(lv_obj_t *obj)`

Move the cursor one line up

**Parametri**

**obj** -- pointer to a text area object

## Variables

`const lv_obj_class_t lv_textarea_class`

### lv\_textarea\_private.h

`struct _lv_textarea_t`

`#include <lv_textarea_private.h>` Data of text area

#### Public Members

`lv_obj_t obj`

`lv_obj_t *label`

Label of the text area

`char *placeholder_txt`

Place holder label, only visible if text is an empty string

`char *pwd_tmp`

Used to store the original text in password mode

**char \*pwd\_bullet**

Replacement characters displayed in password mode

**const char \*accepted\_chars**

Only these characters will be accepted. NULL: accept all

**uint32\_t max\_length**

The max. number of characters. 0: no limit

**uint32\_t pwd\_show\_time**

Time to show characters in password mode before change them to '\*'

**int32\_t valid\_x**

Used when stepping up/down to a shorter line. (Used by the library)

**uint32\_t pos**

The current cursor position (0: before 1st letter; 1: before 2nd letter ...)

***lv\_area\_t* area**

Cursor area relative to the Text Area

**uint32\_t txt\_byte\_pos**

Byte index of the letter after (on) the cursor

**uint8\_t show**

Cursor is visible now or not (Handled by the library)

**uint8\_t click\_pos**

1: Enable positioning the cursor by clicking the text area

**struct *lv\_textarea\_t* cursor****uint32\_t sel\_start**

Temporary values for text selection

**uint32\_t sel\_end****uint8\_t text\_sel\_in\_prog**

User is in process of selecting

**uint8\_t text\_sel\_en**

Text can be selected on this text area

**uint8\_t pwd\_mode**

Replace characters with '\*'

**uint8\_t one\_line**

One line mode (ignore line breaks)

## 14.24.35 tileview

**lv\_tileview.h**

### Functions

**lv\_obj\_t \*lv\_tileview\_create(lv\_obj\_t \*parent)**

Create a tileview object

#### Parametri

- parent** -- pointer to an object, it will be the parent of the new tileview

#### Ritorna

pointer to the created tileview

**lv\_obj\_t \*lv\_tileview\_add\_tile(lv\_obj\_t \*tv, uint8\_t col\_id, uint8\_t row\_id, lv\_dir\_t dir)**

Add a tile to the tileview

#### Parametri

- tv** -- pointer to the tileview object
- col\_id** -- column id of the tile
- row\_id** -- row id of the tile
- dir** -- direction to move to the next tile

#### Ritorna

pointer to the added tile object

**void lv\_tileview\_set\_tile(lv\_obj\_t \*tv, lv\_obj\_t \*tile\_obj, lv\_anim\_enable\_t anim\_en)**

Set the active tile in the tileview.

#### Parametri

- parent** -- pointer to the tileview object
- tile\_obj** -- pointer to the tile object to be set as active
- anim\_en** -- animation enable flag (LV\_ANIM\_ON or LV\_ANIM\_OFF)

**void lv\_tileview\_set\_tile\_by\_index(lv\_obj\_t \*tv, uint32\_t col\_id, uint32\_t row\_id, lv\_anim\_enable\_t anim\_en)**

Set the active tile by index in the tileview

#### Parametri

- tv** -- pointer to the tileview object
- col\_id** -- column id of the tile to be set as active
- row\_id** -- row id of the tile to be set as active
- anim\_en** -- animation enable flag (LV\_ANIM\_ON or LV\_ANIM\_OFF)

**lv\_obj\_t \*lv\_tileview\_get\_tile\_active(lv\_obj\_t \*obj)**

Get the currently active tile in the tileview

#### Parametri

- obj** -- pointer to the tileview object

#### Ritorna

pointer to the currently active tile object

### Variables

const lv\_obj\_class\_t **lv\_tileview\_class**

const lv\_obj\_class\_t **lv\_tileview\_tile\_class**

**lv\_tileview\_private.h**

struct **\_lv\_tileview\_t**

**Public Members**

**lv\_obj\_t obj**

**lv\_obj\_t \*tile\_act**

**struct \_lv\_tileview\_tile\_t**

**Public Members**

**lv\_obj\_t obj**

**lv\_dir\_t dir**

**14.24.36 win**

**lv\_win.h**

**Functions**

**lv\_obj\_t \*lv\_win\_create(lv\_obj\_t \*parent)**

Create a window widget

**Parametri**

- **parent** -- pointer to a parent widget

**Ritorna**

the created window

**lv\_obj\_t \*lv\_win\_add\_title(lv\_obj\_t \*win, const char \*txt)**

Add a title to the window

**Parametri**

- **obj** -- pointer to a window widget
- **txt** -- the text of the title

**Ritorna**

the widget where the content of the title can be created

**lv\_obj\_t \*lv\_win\_add\_button(lv\_obj\_t \*win, const void \*icon, int32\_t btn\_w)**

Add a button to the window

**Parametri**

- **obj** -- pointer to a window widget
- **icon** -- an icon to be displayed on the button
- **btn\_w** -- width of the button

**Ritorna**

the widget where the content of the button can be created

**lv\_obj\_t \*lv\_win\_get\_header(lv\_obj\_t \*win)**

Get the header of the window

**Parametri**

- **win** -- pointer to a window widget

**Ritorna**

the header of the window

**lv\_obj\_t \*lv\_win\_get\_content(lv\_obj\_t \*win)**

Get the content of the window

**Parametri**

- **win** -- pointer to a window widget

**Ritorna**

the content of the window

## Variables

const lv\_obj\_class\_t **lv\_win\_class**

**lv\_win\_private.h**

struct **\_lv\_win\_t**

### Public Members

lv\_obj\_t **obj**