
百问网 LVGL 中文手册 8.1

www.100ask.net

2021 年 12 月 08 日

Contents

1 Introduction (介绍)	2
1.1 Key features (主要特性)	2
1.2 Requirements (配置要求)	3
1.3 License (许可证)	3
1.4 Repository layout (仓库布局)	4
1.5 Release policy (发布策略)	5
1.6 FAQ (常见问题)	7
2 Examples	11
2.1 Get started	11
2.2 Styles	11
2.3 Animations	11
2.4 Events	11
2.5 Layouts	11
2.6 Scrolling	11
2.7 Widgets	11
3 Get started (开始)	59
3.1 Quick overview (快速概览)	60
3.2 Simulator on PC (PC 上的模拟器)	67
3.3 STM32	71
3.4 NXP	71
3.5 Espressif (ESP32)	74
3.6 Arduino	76
3.7 Micropython	78
3.8 NuttX RTOS	81
4 Porting (移植)	85
4.1 Set-up a project (设置项目)	85
4.2 Display interface (显示接口)	86
4.3 Input device interface (输入设备接口)	98
4.4 Tick interface (心跳接口)	108
4.5 Task Handler (任务处理器)	109
4.6 Sleep management (睡眠管理)	110
4.7 Operating system and interrupts (操作系统和中断)	110
4.8 Logging (日志)	111

5 Overview (概览)	113
5.1 Objects (对象)	113
5.2 Positions, sizes, and layouts (位置、大小和布局)	121
5.3 Styles (风格样式)	132
5.4 Style properties	167
5.5 Scroll (滚动)	177
5.6 Layers (图层)	183
5.7 Events (事件)	185
5.8 Input devices (输入设备)	192
5.9 Displays (显示)	201
5.10 Colors (颜色)	208
5.11 Fonts (字体)	217
5.12 Images (图象)	227
5.13 File system (文件系统)	244
5.14 Animations (动画)	252
5.15 Timers (定时器)	263
5.16 Drawing (绘画)	268
5.17 New widget	276
6 Widgets (部件)	277
6.1 Base object (基础对象) (lv_obj)	277
6.2 Core widgets (核心部件)	292
6.3 Extra widgets (附加部件)	420
7 Layouts (布局)	483
7.1 Flex (弹性布局)	483
7.2 Grid (网格布局)	489
8 3rd party libraries(第三方库)	497
8.1 File System Interfaces(文件系统接口)	497
8.2 BMP decoder(BMP 解码器)	498
8.3 JPG decoder(JPG 解码器)	499
8.4 PNG decoder(PNG 解码器)	501
8.5 GIF decoder(GIF 解码器)	501
8.6 FreeType support(支持 FreeType)	502
8.7 QR code(二维码)	504
8.8 Lottie player	505
9 Others (其他)	508
9.1 Snapshot (快照)	508
10 Contributing (贡献)	511
10.1 Introduction (介绍)	511
10.2 Pull request (拉取请求)	512
10.3 Developer Certification of Origin (DCO) (开发者原产地认证 (DCO))	514
10.4 Ways to contribute (贡献方式)	516
11 Changelog (更新日志)	522
11.1 v8.1.0 (In progress) (进行中)	522
11.2 v8.0.2 (16.07.2021)	523
11.3 v8.0.1 (14.06.2021)	525
11.4 v8.0.0 (01.06.2021)	528
11.5 v7.11.0 (16.03.2021)	532
11.6 v7.10.1 (16.02.2021)	532
11.7 v7.10.0 (02.02.2021)	533

11.8 v7.9.1 (19.01.2021)	533
11.9 v7.9.0 (05.01.2021)	533
11.10 v7.8.1 (15.12.2020)	534
11.11 v7.8.0 (01.12.2020)	534
11.12 v7.7.2 (17.11.2020)	535
11.13 v7.7.1 (03.11.2020)	535
11.14 v7.7.0 (20.10.2020)	535
11.15 v7.6.1 (06.10.2020)	536
11.16 v7.6.0 (22.09.2020)	536
11.17 v7.5.0 (15.09.2020)	536
11.18 v7.4.0 (01.09.2020)	537
11.19 v7.3.1 (18.08.2020)	538
11.20 v7.3.0 (04.08.2020)	538
11.21 v7.2.0 (21.07.2020)	539
11.22 v7.1.0 (07.07.2020)	540
11.23 v7.0.2 (16.06.2020)	541
11.24 v7.0.1 (01.06.2020)	541
11.25 v7.0.0 (18.05.2020)	542
12 Roadmap (产品线路)	546
12.1 v8.1	546
12.2 v8.2	548
12.3 Ideas (想法)	548
12.4 v8	549
13 项目实战	551
13.1 在 windwos 模拟器运行 lvgl(v8.0)	551
13.2 STM32F103 LVGL GUI DEMO 效果	556
13.3 STM32MP157 LVGL GUI DEMO 效果	557
13.4 [*] IMX6ULL LVGL GUI DEMO 效果	558
13.5 ESP32 LVGL GUI DEMO 效果	560
14 联系我们	561
15 加入技术交流群聊一起学习!	563
索引	564

PDF version: 100ASK_LVGL_CN.pdf

CHAPTER 1

Introduction (介绍)

LVGL (Light and Versatile Graphics Library) is a free and open-source graphics library providing everything you need to create embedded GUI with easy-to-use graphical elements, beautiful visual effects and low memory footprint.

LVGL(轻量级和通用图形库) 是一个免费和开源的图形库，它提供了创建嵌入式 GUI 所需的一切，具有易于使用的图形元素，美丽的视觉效果和低内存占用。

LVGL 的项目作者是来自匈牙利首都布达佩斯的 *Gábor Kiss-Vámosi*。Kiss 在 2009 年开始写 LVGL(*LittlevGL*)，2016 年将其重写并发布在 [GitHub](#) 上。

LVGL 的第一个版本于 2016 年在 GitHub 上发布，当时叫 *LittlevGL* 而不是 *LVGL*，后来作者统一修改为 *LVGL* 甚至连仓库地址都改了。像一般的开源项目的那样，它是作为一个人的项目开始的。从那时起，陆续有近 100 名贡献者参与了项目开发，使得 *LVGL* 逐渐成为最受欢迎的嵌入式图形库之一。

1.1 Key features (主要特性)

- Powerful building blocks such as buttons, charts, lists, sliders, images etc.
- Advanced graphics with animations, anti-aliasing, opacity, smooth scrolling
- Various input devices such as touchpad, mouse, keyboard, encoder etc.
- Multi-language support with UTF-8 encoding
- Multi-display support, i.e. use more TFT, monochrome displays simultaneously
- Fully customizable graphic elements
- Hardware independent to use with any microcontroller or display
- Scalable to operate with little memory (64 kB Flash, 16 kB RAM)
- OS, External memory and GPU supported but not required
- Single frame buffer operation even with advanced graphical effects

- Written in C for maximal compatibility (C++ compatible)
- Simulator to start embedded GUI design on a PC without embedded hardware
- Binding to MicroPython
- Tutorials, examples, themes for rapid GUI design
- Documentation is available as online and offline
- Free and open-source under MIT license
- 强大的构建块，如按钮，图表，列表，滑块，图像等。
- 高级图形动画，抗锯齿，不透明度，平滑滚动
- 各种输入设备，如触摸板、鼠标、键盘、编码器等
- 多语言支持与 UTF-8 编码
- 多显示器支持，即使用更多的 TFT，单色显示器同时
- 完全可定制的图形元素与 css 类样式
- 硬件独立与任何微控制器或显示器使用
- 可扩展，使用少量内存 (64kb Flash, 16kb RAM)
- 支持操作系统、外部内存和 GPU，但不是必需的
- 单帧缓冲操作，甚至与高级图形效果
- 用 C 编写的最大兼容性 (c++ 兼容)
- 模拟器在没有嵌入式硬件的 PC 上开始嵌入式 GUI 设计
- 绑定到 MicroPython
- 教程，例子，快速 GUI 设计的主题
- 文档可以在线和 PDF 格式获取
- 麻省理工学院许可下的免费和开源

1.2 Requirements (配置要求)

基本上，每个现代控制器（能够驱动显示器）都适合运行 LVGL。最低要求是：

1.3 License (许可证)

The LVGL project (including all repositories) is licensed under [MIT license](#). It means you can use it even in commercial projects.

LVGL 项目（包括所有存储库）在 [MIT license](#) 许可下获得许可。这意味着您甚至可以在商业项目中使用它。It's not mandatory but we highly appreciate it if you write a few words about your project in the [My projects](#) category of the Forum or a private message from [lvgl.io](#).

这不是强制性的，但如果您在论坛的 [My projects](#) 类别或来自 [lvgl.io](#) 的私人消息中写下有关您的项目的几句话，我们将不胜感激。

Although you can get LVGL for free there is a huge work behind it. It's created by a group of volunteers who made it available for you in their free time.

尽管您可以免费获得 LVGL，但它背后的工作量很大。它由一群志愿者创建，他们在空闲时间为您提供。

To make the LVGL project sustainable, please consider [Contributing](#) to the project.

You can choose from [many ways of contributions](#) such as simply writing a tweet about you are using LVGL, fixing bugs, translating the documentation, or even becoming a maintainer.

为了使 LVGL 项目可持续，请考虑为该项目做贡献。您可以从[多种投稿方式](#)中进行选择，例如简单地写一条关于您正在使用 LVGL 的推文、修复错误、翻译文档，甚至成为维护者。

1.4 Repository layout (仓库布局)

All repositories of the LVGL project are hosted on GitHub: <https://github.com/lvgl>

LVGL 项目的所有存储库都托管在 GitHub: <https://github.com/lvgl>

You fill these repositories there:

- [lvgl](#) The library itself
- [lv_examples](#) Examples and demos
- [lv_drivers](#) Display and input device drivers
- [docs](#) Source of the documentation's site (<https://docs.lvgl.io>)
- [blog](#) Source of the blog's site (<https://blog.lvgl.io>)
- [sim](#) Source of the online simulator's site (<https://sim.lvgl.io>)
- [lv_sim_...](#) Simulator projects for various IDEs and platforms
- [lv_port_...](#) LVGL ports to development boards
- [lv_binding_...](#) Bindings to other languages
- [lv_...](#) Ports to other platforms

您可以从面这里获取到所有的仓库:

- [lvgl](#) lvgl 图形库
- [lv_examples](#) 库的示例和演示
- [lv_drivers](#) 显示和输入设备驱动程序
- [docs](#) 文档站点的来源 (<https://docs.lvgl.io>)
- [blog](#) 博客站点的来源 (<https://blog.lvgl.io>)
- [sim](#) 在线模拟器网站的来源 (<https://sim.lvgl.io>)
- [lv_sim_...](#) 适用于各种 IDE 和平台的模拟器项目
- [lv_port_...](#) LVGL 端口到开发板
- [lv_binding_...](#) 与其他语言的绑定
- [lv_...](#) 移植到其他平台

The [lvgl](#), [lv_examples](#) and [lv_drivers](#) are the core repositories which gets the most attentions regarding maintenance.

其中 [lvgl](#), [lv_examples](#) 和 [lv_drivers](#) 是最受维护关注的核心存储库。

1.5 Release policy (发布策略)

The core repositories follow the rules of Semantic versioning:

- Major versions for incompatible API changes. E.g. v5.0.0, v6.0.0
- Minor version for new but backward-compatible functionalities. E.g. v6.1.0, v6.2.0
- Patch version for backward-compatible bug fixes. E.g. v6.1.1, v6.1.2

LVGL 库遵循语义版本管理：

- 不兼容 API 更改的主要版本。比如：v5.0.0, v6.0.0
- 新的但向后兼容的功能的次要版本。比如：v6.1.0, v6.2.0
- 用于向后兼容错误修复的补丁版本。比如：v6.1.1, v6.1.2

1.5.1 Branches (分支)

The core repositories have at least the following branches:

- `master` latest version, patches are merged directly here.
- `dev` merge new features here until they are merged into `master`.
- `release vX` stable versions of the major releases

核心存储库至少有以下分支：

- `master` 最新版本，补丁直接在这里合并。
- `dev` 在此处合并新功能，直到它们合并到 `master` 中。
- `release vX` 主要版本的稳定版本

1.5.2 Release cycle (发布周期)

- Bug fixes: Released on demand even weekly
- Minor releases: Every 3-4 months
- Major releases: Approximately yearly
- 错误修复：每周按需发布
- 次要版本：每 3-4 个月
- 主要版本：大约每年

1.5.3 Tags (版本标签)

The core repositories have at least the following branches:

- `master` latest version, patches are merged directly here.
- `release vX.Y` stable versions of the minor releases
- `fix/some-description` temporary branches for bug fixes
- `feat/some-description` temporary branches for features

核心存储库至少有以下分支：

- `master` 最新版本，补丁直接在这里合并。
- `release/vX.Y` 次要版本的稳定版本
- `fix/some-description` 用于错误修复的临时分支
- `feat/some-description` 用于特性的临时分支

1.5.4 Changelog (变更日志)

The changes are recorded in `CHANGELOG.md`.

更改记录在 `CHANGELOG.md` 中。

1.5.5 Version support

Before v8 every minor release of major releases is supported for 1 year. Starting from v8, every minor release is supported for 1 year.

在 v8 之前，每个主要版本的次要版本都支持 1 年。从 v8 开始，每个次要版本都支持 1 年。

1.5.6 Side projects

The `docs` is rebuilt on every release. By default, the `latest` documentation is displayed which is for the current `master` branch of lvgl. The documentation of earlier versions is available from the menu on the left.

文档 (docs) 仓库会在每个版本上重建。默认情况下，显示当前 lvgl 主 (`master`) 分支的最新 (`latest`) 文档。早期版本的文档可从左侧的菜单中获得。

The simulator, porting, and other projects are updated with best effort. Pull requests are welcome if you updated one of them.

模拟器、移植和其他项目会尽最大努力进行更新。如果您更新其中之一，欢迎请求请求。

1.5.7 Version support (版本支持)

In the core repositories each major version has a branch (e.g. `release/v6`). All the minor and patch releases of that major version are merged there.

在核心存储库中，每个主要版本都有一个分支（例如 `release/v6`）。该主要版本的所有次要版本和补丁版本都合并在那里。

It makes possible to add fixed older versions without bothering the newer ones.

可以在不打扰新版本的情况下添加固定的旧版本。

All major versions are officially supported for 1 year.

所有主要版本都得到官方支持 1 年。

1.6 FAQ (常见问题)

1.6.1 Where can I ask questions? (我可以在哪里提问?)

You can ask questions in the Forum: <https://forum.lvgl.io/>.

可以在论坛提问: <https://forum.lvgl.io/>。

We use [GitHub issues](#) for development related discussion.

So you should use them only if your question or issue is tightly related to the development of the library.

我们使用 [GitHub issues](#) 进行开发相关讨论。因此，仅当您的问题或问题与库的开发密切相关时，才应使用它们。

1.6.2 Is my MCU/hardware supported? (LVGL 是否支持我的 MCU/硬件?)

Every MCU which is capable of driving a display via Parallel port, SPI, RGB interface or anything else and fulfills the [Requirements](#) is supported by LLVGL.

LVGL 支持每个能够通过并行端口、SPI、RGB 接口或其他任何方式驱动显示器并满足要求 的 MCU。

It includes:

- "Common" MCUs like STM32F, STM32H, NXP Kinetis, LPC, iMX, dsPIC33, PIC32 etc.
- Bluetooth, GSM, WiFi modules like Nordic NRF and Espressif ESP32
- Linux frame buffer like /dev/fb0 which includes Single-board computers too like Raspberry Pi
- And anything else with a strong enough MCU and a periphery to drive a display

这包括:

- “常见” 的 MCU，如 STM32F、STM32H、NXP Kinetis、LPC、IMX、dsPIC33、PIC32 等。
- 蓝牙、GSM、WiFi 模块，如 Nordic NRF 和 Espressif ESP32
- Linux 帧缓冲区，如 /dev/fb0，其中也包括单板计算机，如 Raspberry Pi
- 以及任何其他具有足够强大 MCU 和外围设备来驱动显示器的设备

1.6.3 Is my display supported? (支持我的显示器吗?)

LVGL needs just one simple driver function to copy an array of pixels into a given area of the display.

If you can do this with your display then you can use that display with LVGL.

LVGL 只需要一个简单的驱动程序函数即可将像素阵列复制到显示器的给定区域。

如果您可以对显示器执行此操作，那么您可以将该显示器与 LVGL 一起使用。

Some examples of the supported display types:

- TFTs with 16 or 24 bit color depth
- Monitors with HDMI port
- Small monochrome displays
- Gray-scale displays
- even LED matrices

- or any other display where you can control the color/state of the pixels

支持的显示类型的一些示例：

- 具有 16 位或 24 位色深的 TFT
- 带 HDMI 端口的显示器
- 小型单色显示器
- 灰度显示
- 甚至 LED 矩阵
- 或任何其他可以控制像素颜色/状态的显示器

See the [Porting](#) section to learn more.

请参阅[移植](#)部分以了解更多信息。

1.6.4 Nothing happens, my display driver is not called. What have I missed? (没有反应，我的显示驱动程序没有被调用。我错过了什么？)

Be sure you are calling `lv_tick_inc(x)` in an interrupt and `lv_task_handler()` in your main `while(1)`.

确保在中断中调用 `lv_tick_inc(x)`，在主 `while(1)` 中调用 `lv_task_handler()`。

Learn more in the [Tick](#) and [Task handler](#) section.

在[Tick](#) 和[任务处理程序 \(Task handler\)](#) 部分了解更多信息。

1.6.5 Why the display driver is called only once? Only the upper part of the display is refreshed. (为什么显示驱动程序只调用一次？仅刷新显示的上部。)

Be sure you are calling `lv_disp_flush_ready(drv)` at the end of your "display flush callback".

确保在“显示刷新回调”结束时调用 `lv_disp_flush_ready(drv)`。

1.6.6 Why I see only garbage on the screen? (为什么我在屏幕上只看到垃圾？)

Probably there a bug in your display driver. Try the following code without using LVGL. You should see a square with red-blue gradient

您的显示驱动程序中可能存在错误。在不使用 LVGL 的情况下尝试以下代码。你应该看到一个带有红蓝渐变的正方形

```
#define BUF_W 20
#define BUF_H 10

lv_color_t buf[BUF_W * BUF_H];
lv_color_t * buf_p = buf;
uint16_t x, y;
for(y = 0; y < BUF_H; y++) {
    lv_color_t c = lv_color_mix(LV_COLOR_BLUE, LV_COLOR_RED, (y * 255) / BUF_H);
    for(x = 0; x < BUF_W; x++){
        (*buf_p) = c;
        buf_p++;
    }
}
```

(下页继续)

(续上页)

```

}

lv_area_t a;
a.x1 = 10;
a.y1 = 40;
a.x2 = a.x1 + BUF_W - 1;
a.y2 = a.y1 + BUF_H - 1;
my_flush_cb(NULL, &a, buf);

```

1.6.7 Why I see non-sense colors on the screen? (为什么我在屏幕上看到无意义的颜色?)

Probably LVGL's color format is not compatible with your displays color format. Check `LV_COLOR_DEPTH` in `lv_conf.h`.

可能 LVGL 的颜色格式与您的显示器颜色格式不兼容。检查 `lv_conf.h` 中的 `LV_COLOR_DEPTH`。

如果您在 SPI (或其他面向字节的接口) 中使用 16 位颜色, 您可能需要在 `lv_conf.h` 中设置 `LV_COLOR_16_SWAP = 1`。它交换像素的高字节和低字节。

1.6.8 How to speed up my UI? (如何加速我的用户界面?)

- Turn on compiler optimization and enable cache if your MCU has
- Increase the size of the display buffer
- Use 2 display buffers and flush the buffer with DMA (or similar periphery) in the background
- Increase the clock speed of the SPI or Parallel port if you use them to drive the display
- If your display has SPI port consider changing to a model with parallel because it has much higher throughput
- Keep the display buffer in the internal RAM (not in external SRAM) because LVGL uses it a lot and it should have a small access time
- 如果您的 MCU 支持的话, 请打开编译器优化并启用缓存
- 增加显示缓冲区的大小
- 使用 2 个显示缓冲区并在后台使用 DMA (或类似外围设备) 刷新缓冲区
- 如果您使用 SPI 或并行端口来驱动显示器, 请提高它们的时钟速度
- 如果您的显示器具有 SPI 端口, 请考虑更改为并行模型, 因为它具有更高的吞吐量
- 将显示缓冲区保留在内部 RAM (而不是外部 SRAM) 中, 因为 LVGL 经常使用它, 并且访问时间应该很短

1.6.9 How to reduce flash/ROM usage? (如何减少闪存/ROM 的使用?)

You can disable all the unused features (such as animations, file system, GPU etc.) and object types in *lv_conf.h*.

您可以在 *lv_conf.h* 中禁用所有未使用的功能（例如动画、文件系统、GPU 等）和对象类型。

If you are using GCC you can add

- `-fdata-sections -ffunction-sections` compiler flags
- `--gc-sections` linker flag

如果您使用的是 GCC，您可以添加

- `-fdata-sections -ffunction-sections` compiler flags
- `--gc-sections` linker flag

to remove unused functions and variables from the final binary

从最终二进制文件中删除未使用的函数和变量

1.6.10 How to reduce the RAM usage (如何减少内存使用)

- Lower the size of the *Display buffer*
- Reduce `LV_MEM_SIZE` in *lv_conf.h*. This memory used when you create objects like buttons, labels, etc.
- To work with lower `LV_MEM_SIZE` you can create the objects only when required and deleted them when they are not required anymore

降低显示缓冲区的大小减少 *lv_conf.h* 中的 `LV_MEM_SIZE`。创建按钮、标签等对象时使用的内存。要使用较低的 `LV_MEM_SIZE`，您可以仅在需要时创建对象，并在不再需要时删除它们

1.6.11 How to work with an operating system? (如何使用操作系统?)

To work with an operating system where tasks can interrupt each other (preemptive) you should protect LVGL related function calls with a mutex. See the *Operating system and interrupts* section to learn more.

要使用任务可以相互中断（抢占式）的操作系统，您应该使用互斥锁保护与 LVGL 相关的函数调用。请参阅 *操作系统和中断* 部分以了解更多信息。

CHAPTER 2

Examples

2.1 Get started

2.2 Styles

2.3 Animations

2.4 Events

2.5 Layouts

2.5.1 Flex

2.5.2 Grid

2.6 Scrolling

2.7 Widgets

2.7.1 Base object

2.7.2 Arc

Simple Arc

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

#if LV_USE_ARC && LV_BUILD_EXAMPLES

void lv_example_arc_1(void)
{
    /*Create an Arc*/
    lv_obj_t * arc = lv_arc_create(lv_scr_act());
    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, 40);
    lv_obj_center(arc);
}

#endif
```

```
# Create style for the Arcs
style = lv.style_t()
lv.style_copy(style, lv.style_plain)
style.line.color = lv.color_make(0,0,255) # Arc color
style.line.width = 8                      # Arc width

# Create an Arc
arc = lv.arc(lv.scr_act())
arc.set_style(lv.arc.STYLE.MAIN, style)    # Use the new style
arc.set_angles(90, 60)
arc.set_size(150, 150)
arc.align(None, lv.ALIGN.CENTER, 0, 0)
```

Loader with 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(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_scr_act());
    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_clear_flag(arc, LV_OBJ_FLAG_CLICKABLE); /*To not allow adjusting by click*/
}
```

(下页继续)

(续上页)

```

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_time(&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

```

```

# Create an arc which acts as a loader.
class loader_arc(lv.arc):

    def __init__(self, parent, color=lv.color_hex(0x000080),
                 width=8, style=lv.style_plain, rate=20):
        super().__init__(parent)

        self.a = 0
        self.rate = rate

        # Create style for the Arcs
        self.style = lv.style_t()
        lv.style_copy(self.style, style)
        self.style.line.color = color
        self.style.line.width = width

        # Create an Arc
        self.set_angles(180, 180);
        self.set_style(self.STYLE.MAIN, self.style);

        # Spin the Arc
        self.spin()

    def spin(self):
        # Create an `lv_task` to update the arc.
        lv.task_create(self.task_cb, self.rate, lv.TASK_PRIO.LOWEST, {})

        # An `lv_task` to call periodically to set the angles of the arc
    def task_cb(self, task):
        self.a+=5;
        if self.a >= 359: self.a = 359

        if self.a < 180: self.set_angles(180-self.a, 180)
        else: self.set_angles(540-self.a, 180)

        if self.a == 359:
            self.a = 0

```

(下页继续)

(续上页)

```
lv.task_del(task)

# Create a loader arc
loader_arc = loader_arc(lv.scr_act())
loader_arc.align(None, lv.ALIGN.CENTER, 0, 0)
```

2.7.3 Bar

Simple Bar

```
#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.scr_act());
    lv_obj_set_size(bar1, 200, 20);
    lv_obj_center(bar1);
    lv_bar_set_value(bar1, 70, LV_ANIM_OFF);
}

#endif
```

```
bar1 = lv.bar(lv.scr_act())
bar1.set_size(200, 30)
bar1.align(None, lv.ALIGN.CENTER, 0, 0)
bar1.set_anim_time(1000)
bar1.set_value(100, lv.ANIM.ON)
```

Styling a bar

```
#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_time(&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));
```

(下页继续)

(续上页)

```

lv_style_set_radius(&style_indic, 3);

lv_obj_t * bar = lv_bar_create(lv_scr_act());
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

```

Error encountered **while** trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/bar/lv_example_bar_2.py

Temperature meter

```

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

static void set_temp(void * bar, int32_t temp)
{
    lv_bar_set_value(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_scr_act());
    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_time(&a, 3000);
    lv_anim_set_playback_time(&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);
}

```

(下页继续)

(续上页)

```

    lv_anim_start(&a);
}

#endif

```

Error encountered **while** trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/bar/lv_example_bar_3.py

Stripe pattern and range value

```

#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_IMG_DECLARE(img_skew_strip);
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_img_src(&style_indic, &img_skew_strip);
    lv_style_set_bg_img_tiled(&style_indic, true);
    lv_style_set_bg_img_opa(&style_indic, LV_OPA_30);

    lv_obj_t * bar = lv_bar_create(lv_scr_act());
    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

```

Error encountered **while** trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/bar/lv_example_bar_4.py

Bar with RTL and RTL base direction

```

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

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

```

(下页继续)

(续上页)

```

lv_obj_t * label;

lv_obj_t * bar_ltr = lv_bar_create(lv_scr_act());
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_scr_act());
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_scr_act());
lv_obj_set_base_dir(bar_rtl, LV_BIDI_DIR_RTL);
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_scr_act());
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

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/bar/lv_example_bar_5.py

Custom drawr to show the current value

```

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

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

static void event_cb(lv_event_t * e)
{
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
    if(dsc->part != LV_PART_INDICATOR) return;

    lv_obj_t * obj= lv_event_get_target(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", lv_bar_get_value(obj));

    lv_point_t txt_size;
    lv_txt_get_size(&txt_size, buf, label_dsc.font, label_dsc.letter_space, label_dsc.
    ↵line_space, LV_COORD_MAX, label_dsc.flag);

```

(下页继续)

(续上页)

```

lv_area_t txt_area;
/*If the indicator is long enough put the text inside on the right*/
if(lv_area_get_width(dsc->draw_area) > txt_size.x + 20) {
    txt_area.x2 = dsc->draw_area->x2 - 5;
    txt_area.x1 = txt_area.x2 - txt_size.x + 1;
    label_dsc.color = lv_color_white();
}
/*If the indicator is still short put the text out of it on the right*/
else {
    txt_area.x1 = dsc->draw_area->x2 + 5;
    txt_area.x2 = txt_area.x1 + txt_size.x - 1;
    label_dsc.color = lv_color_black();
}

txt_area.y1 = dsc->draw_area->y1 + (lv_area_get_height(dsc->draw_area) - txt_size.
y) / 2;
txt_area.y2 = txt_area.y1 + txt_size.y - 1;

lv_draw_label(&txt_area, dsc->clip_area, &label_dsc, buf, NULL);
}

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

    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_time(&a, 2000);
    lv_anim_set_playback_time(&a, 2000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}

#endif

```

Error encountered **while** trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/bar/lv_example_bar_6.py

2.7.4 Button

Simple Buttons

```
#include "../../lv_examples.h"
#if LV_USE_BTN && 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_btn_1(void)
{
    lv_obj_t * label;

    lv_obj_t * btn1 = lv_btn_create(lv_scr_act());
    lv_obj_add_event_cb(btn1, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -40);

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

    lv_obj_t * btn2 = lv_btn_create(lv_scr_act());
    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
```

```
def event_handler(obj, event):
    if event == lv.EVENT.CLICKED:
        print("Clicked")

btn1 = lv.btn(lv.scr_act())
btn1.set_event_cb(event_handler)
btn1.align(None, lv.ALIGN.CENTER, 0, -40)

label = lv.label(btn1)
label.set_text("Button")

btn2 = lv.btn(lv.scr_act())
# callback can be lambda:
btn2.set_event_cb(lambda obj, event: print("Toggled") if event == lv.EVENT.VALUE_
    ↪CHANGED else None)
```

(下页继续)

(续上页)

```
btn2.align(None, lv.ALIGN.CENTER, 0, 40)
btn2.set_toggle(True)
btn2.toggle()
btn2.set_fit2(lv.FIT.NONE, lv.FIT.TIGHT)

label = lv.label(btn2)
label.set_text("Toggled")
```

Styling buttons

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

/***
 * Style a button from scratch
 */
void lv_example_btn_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_VERT);

    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_ofs_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);

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

    lv_style_set_translate_y(&style_pr, 5);
    lv_style_set_shadow_ofs_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));
```

(下页继续)

(续上页)

```

/*Add a transition to the 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_btn_create(lv_scr_act());
lv_obj_remove_style_all(btn1);                                /*Remove the style coming
→ from the theme*/
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

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets btn/lv_example_btn_2.py

Gummy button

```

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

/**
 * Create a style transition on a button to act like a gum when clicked
 */
void lv_example_btn_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);

```

(下页继续)

(续上页)

```

/*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_btn_create(lv_scr_act());
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

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/button/lv_example_button_3.py

2.7.5 Button matrix

Simple Button matrix

```

#include "../lv_examples.h"
#if LV_USE_BTNMATRIX && 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(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        uint32_t id = lv_btnmatrix_get_selected_btn(obj);
        const char * txt = lv_btnmatrix_get_btn_text(obj, id);

        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_btnmatrix_1(void)
{
    lv_obj_t * bnm1 = lv_btnmatrix_create(lv_scr_act());

```

(下页继续)

(续上页)

```

lv_btmatrix_set_map(btnm1, btm_map);
lv_btmatrix_set_btn_width(btnm1, 10, 2);           /*Make "Action1" twice as wide as "Action2"*/
lv_btmatrix_set_btn_ctrl(btnm1, 10, LV_BTNMATRIX_CTRL_CHECKABLE);
lv_btmatrix_set_btn_ctrl(btnm1, 11, LV_BTNMATRIX_CTRL_CHECKED);
lv_obj_align(btnm1, LV_ALIGN_CENTER, 0, 0);
lv_obj_add_event_cb(btnm1, event_handler, LV_EVENT_ALL, NULL);
}

#endif

```

```

def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
        txt = obj.get_active_btn_text()
        print("%s was pressed" % txt)

btm_map = ["1", "2", "3", "4", "5", "\n",
           "6", "7", "8", "9", "0", "\n",
           "Action1", "Action2", ""]

btnm1 = lv.btnm(lv.scr_act())
btnm1.set_map(btm_map)
btnm1.set_btn_width(10, 2)           # Make "Action1" twice as wide as "Action2"
btnm1.align(None, lv.ALIGN.CENTER, 0, 0)
btnm1.set_event_cb(event_handler)

```

Custom buttons

```

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

static void event_cb(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_DRAW_PART_BEGIN) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);

        /*Change the draw descriptor the 2nd button*/
        if(dsc->id == 1) {
            dsc->rect_dsc->radius = 0;
            if(lv_btmatrix_get_selected_btn(obj) == dsc->id)  dsc->rect_dsc->bg_color = lv_palette_darken(LV_PALETTE_GREY, 3);
            else dsc->rect_dsc->bg_color = lv_palette_main(LV_PALETTE_BLUE);

            dsc->rect_dsc->shadow_width = 6;
            dsc->rect_dsc->shadow_ofs_x = 3;
            dsc->rect_dsc->shadow_ofs_y = 3;
            dsc->label_dsc->color = lv_color_white();
        }
        /*Change the draw descriptor the 3rd button*/
        else if(dsc->id == 2) {
            dsc->rect_dsc->radius = LV_RADIUS_CIRCLE;
        }
    }
}

```

(下页继续)

(续上页)

```

    if(lv_btnmatrix_get_selected_btn(obj) == dsc->id)  dsc->rect_dsc->bg_
    ↵color = lv_palette_darken(LV_PALETTE_RED, 3);
    else dsc->rect_dsc->bg_color = lv_palette_main(LV_PALETTE_RED);

    dsc->label_dsc->color = lv_color_white();
}
else if(dsc->id == 3) {
    dsc->label_dsc->opa = LV_OPA_TRANSP; /*Hide the text if any*/
}

if(code == LV_EVENT_DRAW_PART_END) {
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);

    /*Add custom content to the 4th button when the button itself was drawn*/
    if(dsc->id == 3) {
        LV_IMG_DECLARE(img_star);
        lv_img_header_t header;
        lv_res_t res = lv_img_decoder_get_info(&img_star, &header);
        if(res != LV_RES_OK) return;

        lv_area_t a;
        a.x1 = dsc->draw_area->x1 + (lv_area_get_width(dsc->draw_area) - header.
        ↵w) / 2;
        a.x2 = a.x1 + header.w - 1;
        a.y1 = dsc->draw_area->y1 + (lv_area_get_height(dsc->draw_area) - header.
        ↵h) / 2;
        a.y2 = a.y1 + header.h - 1;

        lv_draw_img_dsc_t img_draw_dsc;
        lv_draw_img_dsc_init(&img_draw_dsc);
        img_draw_dsc.recolor = lv_color_black();
        if(lv_btnmatrix_get_selected_btn(obj) == dsc->id) img_draw_dsc.recolor_
        ↵opa = LV_OPA_30;

        lv_draw_img(&a, dsc->clip_area, &img_star, &img_draw_dsc);
    }
}
}

/**
 * Add custom drawer to the button matrix to customize buttons one by one
 */
void lv_example_btnmatrix_2(void)
{
    lv_obj_t * btnm = lv_btnmatrix_create(lv_scr_act());
    lv_obj_add_event_cb(btnm, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_center(btnm);
}

#endif

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/btnmatrix/lv_example_btnmatrix_2.py

Pagination

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

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    uint32_t id = lv_btnmatrix_get_selected_btn(obj);
    bool prev = id == 0 ? true : false;
    bool next = id == 6 ? true : false;
    if(prev || next) {
        /*Find the checked button*/
        uint32_t i;
        for(i = 1; i < 7; i++) {
            if(lv_btnmatrix_has_btn_ctrl(obj, i, LV_BTNMATRIX_CTRL_CHECKED)) break;
        }

        if(prev && i > 1) i--;
        else if(next && i < 5) i++;

        lv_btnmatrix_set_btn_ctrl(obj, i, LV_BTNMATRIX_CTRL_CHECKED);
    }
}

/**
 * Make a button group (pagination)
 */
void lv_example_btnmatrix_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_btnmatrix_create(lv_scr_act());
    lv_btnmatrix_set_map(btnm, map);
    lv_obj_add_style(btnm, &style_bg, 0);
    lv_obj_add_style(btnm, &style_btn, LV_PART_ITEMS);
    lv_obj_add_event_cb(btnm, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_size(btnm, 225, 35);
}
```

(下页继续)

(续上页)

```

/*Allow selecting on one number at time*/
lv_btnmatrix_set_btn_ctrl_all(btnm, LV_BTNMATRIX_CTRL_CHECKABLE);
lv_btnmatrix_clear_btn_ctrl(btnm, 0, LV_BTNMATRIX_CTRL_CHECKABLE);
lv_btnmatrix_clear_btn_ctrl(btnm, 6, LV_BTNMATRIX_CTRL_CHECKABLE);

lv_btnmatrix_set_one_checked(btnm, true);
lv_btnmatrix_set_btn_ctrl(btnm, 1, LV_BTNMATRIX_CTRL_CHECKED);

lv_obj_center(btnm);

}

#endif

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/buttonmatrix/lv_example_buttonmatrix_3.py

2.7.6 Calendar

Calendar with header

```

#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_event_get_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_scr_act());
    lv_obj_set_size(calendar, 200, 200);
    lv_obj_align(calendar, LV_ALIGN_CENTER, 0, 20);
    lv_obj_add_event_cb(calendar, event_handler, LV_EVENT_ALL, NULL);

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

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

```

(下页继续)

(续上页)

```

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_header_dropdown_create(lv_scr_act(), calendar);
#elif LV_USE_CALENDAR_HEADER_ARROW
    lv_calendar_header_arrow_create(lv_scr_act(), calendar, 25);
#endif
}

#endif

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/calendar/lv_example_calendar_1.py

2.7.7 Canvas

Drawing on the Canvas and rotate

```

#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_HOR;
    rect_dsc.bg_color = lv_palette_main(LV_PALETTE_RED);
    rect_dsc.bg_grad_color = lv_palette_main(LV_PALETTE_BLUE);
    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_ofs_x = 5;
    rect_dsc.shadow_ofs_y = 5;

    lv_draw_label_dsc_t label_dsc;
    lv_draw_label_dsc_init(&label_dsc);
    label_dsc.color = lv_palette_main(LV_PALETTE_YELLOW);
}

```

(下页继续)

(续上页)

```

static lv_color_t cbuf[LV_CANVAS_BUF_SIZE_TRUE_COLOR(CANVAS_WIDTH, CANVAS_
HEIGHT)];;

lv_obj_t * canvas = lv_canvas_create(lv_scr_act());
lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_IMG_CF_TRUE_
COLOR);
lv_obj_center(canvas);
lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);

lv_canvas_draw_rect(canvas, 70, 60, 100, 70, &rect_dsc);

lv_canvas_draw_text(canvas, 40, 20, 100, &label_dsc, "Some text on text canvas");

/*Test the rotation. It requires an other buffer where the original image is_
stored.
*So copy the current image to buffer and rotate it to the canvas*/
static lv_color_t cbuf_tmp[CANVAS_WIDTH * CANVAS_HEIGHT];
memcpy(cbuf_tmp, cbuf, sizeof(cbuf_tmp));
lv_img_dsc_t img;
img.data = (void *)cbuf_tmp;
img.header.cf = LV_IMG_CF_TRUE_COLOR;
img.header.w = CANVAS_WIDTH;
img.header.h = CANVAS_HEIGHT;

lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);
lv_canvas_transform(canvas, &img, 30, LV_IMG_ZOOM_NONE, 0, 0, CANVAS_WIDTH / 2,_
CANVAS_HEIGHT / 2, true);
}

#endif

```

```

CANVAS_WIDTH = 200
CANVAS_HEIGHT = 150

style = lv.style_t()
lv.style_copy(style, lv.style_plain)
style.body.main_color = lv.color_make(0xFF,0,0)
style.body.grad_color = lv.color_make(0x80,0,0)
style.body.radius = 4
style.body.border.width = 2
style.body.border.color = lv.color_make(0xFF,0xFF,0xFF)
style.body.shadow.color = lv.color_make(0xFF,0xFF,0xFF)
style.body.shadow.width = 4
style.line.width = 2
style.line.color = lv.color_make(0,0,0)
style.text.color = lv.color_make(0,0,0xFF)

# CF.TRUE_COLOR requires 4 bytes per pixel
cbuf = bytearray(CANVAS_WIDTH * CANVAS_HEIGHT * 4)

canvas = lv.canvas(lv.scr_act())
canvas.set_buffer(cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, lv.img.CF.TRUE_COLOR)
canvas.align(None, lv.ALIGN.CENTER, 0, 0)
canvas.fill_bg(lv.color_make(0xC0, 0xC0, 0xC0))

```

(下页继续)

(续上页)

```
canvas.draw_rect(70, 60, 100, 70, style)

canvas.draw_text(40, 20, 100, style, "Some text on text canvas", lv.label.ALIGN.LEFT)

# Test the rotation. It requires an other buffer where the original image is stored.
# So copy the current image to buffer and rotate it to the canvas
img = lv.img_dsc_t()
img.data = cbuf[:]
img.header.cf = lv.img.CF.TRUE_COLOR
img.header.w = CANVAS_WIDTH
img.header.h = CANVAS_HEIGHT

canvas.fill_bg(lv.color_make(0xC0, 0xC0, 0xC0))
canvas.rotate(img, 30, 0, 0, CANVAS_WIDTH // 2, CANVAS_HEIGHT // 2)
```

Transparent Canvas with chroma keying

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

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/**
 * Create a transparent canvas with Chroma keying and indexed color format (palette).
 */
void lv_example_canvas_2(void)
{
    /*Create a button to better see the transparency*/
    lv_btn_create(lv_scr_act());

    /*Create a buffer for the canvas*/
    static lv_color_t cbuf[LV_CANVAS_BUF_SIZE_INDEXED_1BIT(CANVAS_WIDTH, CANVAS_
    ↵HEIGHT)];

    /*Create a canvas and initialize its the palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_scr_act());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_IMG_CF_INDEXED_
    ↵1BIT);
    lv_canvas_set_palette(canvas, 0, LV_COLOR_CHROMA_KEY);
    lv_canvas_set_palette(canvas, 1, lv_palette_main(LV_PALETTE_RED));

    /*Create colors with the indices of the palette*/
    lv_color_t c0;
    lv_color_t c1;

    c0.full = 0;
    c1.full = 1;

    /*Red background (There is no dedicated alpha channel in indexed images so LV_OPA_
    ↵COVER is ignored)*/
    lv_canvas_fill_bg(canvas, c1, LV_OPA_COVER);

    /*Create hole on the canvas*/
    uint32_t x;
```

(下页继续)

(续上页)

```

uint32_t y;
for( y = 10; y < 30; y++) {
    for( x = 5; x < 20; x++) {
        lv_canvas_set_px(canvas, x, y, c0);
    }
}

}

#endif

```

```

# Create a transparent canvas with Chroma keying and indexed color format (palette).

CANVAS_WIDTH  = 50
CANVAS_HEIGHT = 50

def bufsize(w, h, bits, indexed=False):
    """this function determines required buffer size
       depending on the color depth"""
    size = (w * bits // 8 + 1) * h
    if indexed:
        # + 4 bytes per palette color
        size += 4 * (2**bits)
    return size

# Create a button to better see the transparency
lv.btn(lv.scr_act())

# Create a buffer for the canvas
cbuf = bytearray(bufsize(CANVAS_WIDTH, CANVAS_HEIGHT, 1, indexed=True))

# Create a canvas and initialize its the palette
canvas = lv.canvas(lv.scr_act())
canvas.set_buffer(cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, lv.img.CF.INDEXED_1BIT)
# transparent color can be defined in lv_conf.h and set to pure green by default
canvas.set_palette(0, lv.color_make(0x00, 0xFF, 0x00))
canvas.set_palette(1, lv.color_make(0xFF, 0x00, 0x00))

# Create colors with the indices of the palette
c0 = lv.color_t()
c1 = lv.color_t()

c0.full = 0
c1.full = 1

# Transparent background
canvas.fill_bg(c1)

# Create hole on the canvas
for y in range(10,30):
    for x in range(5, 20):
        canvas.set_px(x, y, c0)

```

2.7.8 Chart

2.7.9 Checkbox

Simple Checkboxes

```
#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(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        const char * txt = lv_checkbox_get_text(obj);
        const char * state = lv_obj_get_state(obj) & LV_STATE_CHECKED ? "Checked" :
        ↪"Unchecked";
        LV_LOG_USER("%s: %s", txt, state);
    }
}

void lv_example_checkbox_1(void)
{
    lv_obj_set_flex_flow(lv_scr_act(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_scr_act(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_
    ↪FLEX_ALIGN_CENTER);

    lv_obj_t * cb;
    cb = lv_checkbox_create(lv_scr_act());
    lv_checkbox_set_text(cb, "Apple");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_scr_act());
    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_scr_act());
    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_scr_act());
    lv_obj_add_state(cb, LV_STATE_CHECKED | 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
```

```
def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
        print("State: %s" % ("Checked" if obj.is_checked() else "Unchecked"))
```

(下页继续)

(续上页)

```
cb = lv.cb(lv.scr_act())
cb.set_text("I agree to terms and conditions.")
cb.align(None, lv.ALIGN.CENTER, 0, 0)
cb.set_event_cb(event_handler)
```

2.7.10 Colorwheel

2.7.11 Dropdown

Simple Drop down list

```
#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(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_scr_act());
    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
```

```
def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
        option = " "*10 # should be large enough to store the option
        obj.get_selected_str(option, len(option))
        # .strip() removes trailing spaces
        print("Option: \"%s\" % option.strip())
```

(下页继续)

(续上页)

```
# Create a drop down list
ddlist = lv.ddlist(lv.scr_act())
ddlist.set_options("\n".join([
    "Apple",
    "Banana",
    "Orange",
    "Melon",
    "Grape",
    "Raspberry"]))
ddlist.set_fix_width(150)
ddlist.set_draw_arrow(True)
ddlist.align(None, lv.ALIGN.IN_TOP_MID, 0, 20)
ddlist.set_event_cb(event_handler)
```

Drop down in four directions

```
#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\n"
                               "Grape\n"
                               "Raspberry";

    lv_obj_t * dd;
    dd = lv_dropdown_create(lv_scr_act());
    lv_dropdown_set_options_static(dd, opts);
    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 10);

    dd = lv_dropdown_create(lv_scr_act());
    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_scr_act());
    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_scr_act());
    lv_dropdown_set_options_static(dd, opts);
    lv_dropdown_set_dir(dd, LV_DIR_LEFT);
    lv_dropdown_set_symbol(dd, LV_SYMBOL_LEFT);
```

(下页继续)

(续上页)

```

    lv_obj_align(dd, LV_ALIGN_RIGHT_MID, -10, 0);
}

#endif

```

```

# Create a drop UP list by applying auto realign

# Create a drop down list
ddlist = lv_ddlist(lv.scr_act())
ddlist.set_options("\n".join([
    "Apple",
    "Banana",
    "Orange",
    "Melon",
    "Grape",
    "Raspberry"]))

ddlist.set_fix_width(150)
ddlist.set_fix_height(150)
ddlist.set_draw_arrow(True)

# Enable auto-realign when the size changes.
# It will keep the bottom of the ddlist fixed
ddlist.set_auto_realign(True)

# It will be called automatically when the size changes
ddlist.align(None, lv.ALIGN.IN_BOTTOM_MID, 0, -20)

```

Menu

```

#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(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_scr_act());
    lv_obj_align(dropdown, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_dropdown_set_options(dropdown, "New project\n"
                            "New file\n"
                            "Open project\n"

```

(下页继续)

(续上页)

```

"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_IMG_DECLARE(img_caret_down)
lv_dropdown_set_symbol(dropdown, &img_caret_down);
lv_obj_set_style_transform_angle(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

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_ lvgl_docs_8.x/examples/widgets/dropdown/lv_example_dropdown_3.py

2.7.12 Image

Image from variable and symbol

```

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

void lv_example_img_1(void)
{
    LV_IMG_DECLARE(img_cogwheel_argb);
    lv_obj_t * img1 = lv_img_create(lv_scr_act());
    lv_img_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_CENTER, 0, -20);
    lv_obj_set_size(img1, 200, 200);

    lv_obj_t * img2 = lv_img_create(lv_scr_act());
    lv_img_set_src(img2, LV_SYMBOL_OK "Accept");
    lv_obj_align_to(img2, img1, LV_ALIGN_OUT_BOTTOM_MID, 0, 20);
}

#endif

```

```

from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

```

(下页继续)

(续上页)

```
# Create a screen with a dragable image

with open('cogwheel.png','rb') as f:
    png_data = f.read()

png_img_dsc = lv.img_dsc_t({
    'data_size': len(png_data),
    'data': png_data
})

scr = lv.scr_act()

# Create an image on the left using the decoder

# lv.img.cache_set_size(2)
img1 = lv.img(scr)
img1.align(scr, lv.ALIGN.CENTER, 0, -20)
img1.set_src(png_img_dsc)

img2 = lv.img(scr)
img2.set_src(lv.SYMBOL.OK + "Accept")
img2.align(img1, lv.ALIGN.OUT_BOTTOM_MID, 0, 20)
```

Image recoloring

```
#include "../../lv_examples.h"
#if LV_USE_IMG && 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_img_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);
```

(下页继续)

(续上页)

```

lv_obj_align_to(intense_slider, blue_slider, LV_ALIGN_OUT_RIGHT_MID, 25, 0);

/*Now create the actual image*/
LV_IMG_DECLARE(img_cogwheel_argb)
img1 = lv_img_create(lv_scr_act());
lv_img_set_src(img1, &img_cogwheel_argb);
lv_obj_align(img1, LV_ALIGN_RIGHT_MID, -20, 0);

lv_event_send(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(lv_slider_get_value(red_slider), lv_slider_get_
→value(green_slider), lv_slider_get_value(blue_slider));
    lv_opa_t intense = lv_slider_get_value(intense_slider);
    lv_obj_set_style_img_recolor_opa(img1, intense, 0);
    lv_obj_set_style_img_recolor(img1, color, 0);
}

static lv_obj_t * create_slider(lv_color_t color)
{
    lv_obj_t * slider = lv_slider_create(lv_scr_act());
    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

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
→lvgl_docs_8.x/examples/widgets/img/lv_example_img_2.py

Rotate and zoom

```

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

static void set_angle(void * img, int32_t v)
{
    lv_img_set_angle(img, v);
}

static void set_zoom(void * img, int32_t v)
{
    lv_img_set_zoom(img, v);
}

```

(下页继续)

(续上页)

```
/*
 * Show transformations (zoom and rotation) using a pivot point.
 */
void lv_example_img_3(void)
{
    LV_IMG_DECLARE(img_cogwheel_argb);

    /*Now create the actual image*/
    lv_obj_t * img = lv_img_create(lv_scr_act());
    lv_img_set_src(img, &img_cogwheel_argb);
    lv_obj_align(img, LV_ALIGN_CENTER, 50, 50);
    lv_img_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_time(&a, 5000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, set_zoom);
    lv_anim_set_values(&a, 128, 256);
    lv_anim_set_playback_time(&a, 3000);
    lv_anim_start(&a);
}

#endif
```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/img/lv_example_img_3.py

Image offset and styling

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

static void ofs_y_anim(void * img, int32_t v)
{
    lv_img_set_offset_y(img, v);
}

/**
 * Image styling and offset
 */
void lv_example_img_4(void)
{
    LV_IMG_DECLARE(img_skew_strip);

    static lv_style_t style;
```

(下页继续)

(续上页)

```

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_img_recolor_opa(&style, LV_OPA_COVER);
lv_style_set_img_recolor(&style, lv_color_black());

lv_obj_t * img = lv_img_create(lv_scr_act());
lv_obj_add_style(img, &style, 0);
lv_img_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);
lv_anim_set_time(&a, 3000);
lv_anim_set_playback_time(&a, 500);
lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
lv_anim_start(&a);

}

#endif

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/img/lv_example_img_4.py

2.7.13 Image button

2.7.14 Keyboard

2.7.15 Label

Line wrap, recoloring and scrolling

```

#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_scr_act());
    lv_label_set_long_mode(label1, LV_LABEL_LONG_WRAP);           /*Break the long lines*/
    lv_label_set_recolor(label1, true);                            /*Enable re-coloring by*/
    lv_label_set_text(label1, "#0000ff Re-color# #ff00ff words# #ff0000 of a# label,*/
    lv_label_set_align(label1, LV_ALIGN_CENTER);                  /*align the lines to the center*/
    lv_label_set_text(label1, "and wrap long text automatically.");
    lv_obj_set_width(label1, 150); /*Set smaller width to make the lines wrap*/
}

```

(下页继续)

(续上页)

```

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_scr_act());
lv_label_set_long_mode(label2, LV_LABEL_LONG_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

```

```

label1 = lv.label(lv.scr_act())
label1.set_long_mode(lv.label.LONG.BREAK)          # Break the long lines
label1.set_recolor(True)                          # Enable re-coloring by commands in the text
label1.set_align(lv.label.ALIGN.CENTER)           # Center aligned lines
label1.set_text("#000080 Re-color# #0000ff words# #6666ff of a# label " +
              "and wrap long text automatically.")
label1.set_width(150)
label1.align(None, lv.ALIGN.CENTER, 0, -30)

label2 = lv.label(lv.scr_act())
label2.set_long_mode(lv.label.LONG.SROLL_CIRC)    # Circular scroll
label2.set_width(150)
label2.set_text("It is a circularly scrolling text. ")
label2.align(None, lv.ALIGN.CENTER, 0, 30)

```

Text shadow

```

#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_scr_act());
    lv_obj_add_style(shadow_label, &style_shadow, 0);

    /*Create the main label*/
    lv_obj_t * main_label = lv_label_create(lv_scr_act());
    lv_label_set_text(main_label, "A simple method to create\n" +
                     "shadows on a text.\n")
}

```

(下页继续)

(续上页)

```

        "It even works with\n\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

```

```

# Create a style for the shadow
label_style = lv.style_t()
lv.style_copy(label_style, lv.style_plain)
label_style.text.opa = lv.OPA._50

# Create a label for the shadow first (it's in the background)
shadow_label = lv.label(lv.scr_act())
shadow_label.set_style(lv.label.STYLE.MAIN, label_style)

# Create the main label
main_label = lv.label(lv.scr_act())
main_label.set_text("A simple method to create\n" +
                    "shadows on text\n" +
                    "It even works with\n\n" +
                    "newlines      and spaces.")

# Set the same text for the shadow label
shadow_label.set_text(main_label.get_text())

# Position the main label
main_label.align(None, lv.ALIGN.CENTER, 0, 0)

# Shift the second label down and to the right by 1 pixel
shadow_label.align(main_label, lv.ALIGN.IN_TOP_LEFT, 1, 1)

```

2.7.16 LED

2.7.17 Line

Simple Line

```

#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_t line_points[] = { {5, 5}, {70, 70}, {120, 10}, {180, 60}, {240, u
→10} };

```

(下页继续)

(续上页)

```

/*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_scr_act());
lv_line_set_points(line1, line_points, 5);      /*Set the points*/
lv_obj_add_style(line1, &style_line, 0);
lv_obj_center(line1);
}

#endif

```

```

# Create an array for the points of the line
line_points = [ {"x":5, "y":5},
                {"x":70, "y":70},
                {"x":120, "y":10},
                {"x":180, "y":60},
                {"x":240, "y":10}]

# Create new style (thick dark blue)
style_line = lv.style_t()
lv.style_copy(style_line, lv.style_plain)
style_line.line.color = lv.color_make(0x00, 0x3b, 0x75)
style_line.line.width = 3
style_line.line.rounded = 1

# Copy the previous line and apply the new style
line1 = lv.line(lv.scr_act())
line1.set_points(line_points, len(line_points))      # Set the points
line1.set_style(lv.line.STYLE.MAIN, style_line)
line1.align(None, lv.ALIGN.CENTER, 0, 0)

```

2.7.18 List

2.7.19 Meter

2.7.20 Message box

2.7.21 Roller

Simple Roller

```

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

static void event_handler(lv_event_t * e)

```

(下页继续)

(续上页)

```

{
    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 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_scr_act());
    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

```

```

def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
        option = " "*10
        obj.get_selected_str(option, len(option))
        print("Selected month: %s" % option.strip())

roller1 = lv.roller(lv.scr_act())
roller1.set_options("\n".join([
    "January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
])

```

(下页继续)

(续上页)

```

    "November",
    "December"]),
    lv.roller.MODE.INFINITE)

roller1.set_visible_row_count(4)
roller1.align(None, lv.ALIGN.CENTER, 0, 0)
roller1.set_event_cb(event_handler)

```

Styling the roller

```

#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(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);

    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_scr_act());
    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_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_scr_act());
    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);
}

```

(下页继续)

(续上页)

```

/*A roller on the right with right aligned text, and custom width*/
roller = lv_roller_create(lv_scr_act());
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

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/roller/lv_example_roller_2.py

2.7.22 Slider

Simple Slider

```

#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_scr_act());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_scr_act());
    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(e);
    char buf[8];
    lv_snprintf(buf, sizeof(buf), "%d%%", 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);
}

```

(下页继续)

(续上页)

#endif

```

def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
        print("Value: %d" % obj.get_value())

# Create styles
style_bg = lv.style_t()
style_indic = lv.style_t()
style_knob = lv.style_t()

lv.style_copy(style_bg, lv.style_pretty)
style_bg.body.main_color = lv.color_make(0,0,0)
style_bg.body.grad_color = lv.color_make(0x80, 0x80, 0x80)
style_bg.body.radius = 800 # large enough to make a circle
style_bg.body.border.color = lv.color_make(0xff,0xff,0xff)

lv.style_copy(style_indic, lv.style_pretty_color)
style_indic.body.radius = 800
style_indic.body.shadow.width = 8
style_indic.body.shadow.color = style_indic.body.main_color
style_indic.body.padding.left = 3
style_indic.body.padding.right = 3
style_indic.body.padding.top = 3
style_indic.body.padding.bottom = 3

lv.style_copy(style_knob, lv.style_pretty)
style_knob.body.radius = 800
style_knob.body.opa = lv.OPA._70
style_knob.body.padding.top = 10
style_knob.body.padding.bottom = 10

# Create a slider
slider = lv.slider(lv.scr_act())
slider.set_style(lv.slider.STYLE.BG, style_bg)
slider.set_style(lv.slider.STYLE.INDIC, style_indic)
slider.set_style(lv.slider.STYLE.KNOB, style_knob)
slider.align(None, lv.ALIGN.CENTER, 0, 0)
slider.set_event_cb(event_handler)

```

Slider with custom style

```

#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*/

```

(下页继续)

(续上页)

```

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(0xbabb));
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_scr_act());
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

```

```

def slider_event_cb(slider, event):
    if event == lv.EVENT.VALUE_CHANGED:
        slider_label.set_text("%u" % slider.get_value())

# Create a slider in the center of the display

```

(下页继续)

(续上页)

```

slider = lv.slider(lv.scr_act())
slider.set_width(200)
slider.align(None, lv.ALIGN.CENTER, 0, 0)
slider.set_event_cb(slider_event_cb)
slider.set_range(0, 100)

# Create a label below the slider
slider_label = lv.label(lv.scr_act())
slider_label.set_text("0")
slider_label.set_auto_realign(True)
slider_label.align(slider, lv.ALIGN.OUT_BOTTOM_MID, 0, 10)

# Create an informative label
info = lv.label(lv.scr_act())
info.set_text("""Welcome to the slider+label demo!
Move the slider and see that the label
updates to match it.""")
info.align(None, lv.ALIGN.IN_TOP_LEFT, 10, 10)

```

Slider with extended drawer

```

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

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_scr_act());
    lv_obj_center(slider);

    lv_slider_set_mode(slider, LV_SLIDER_MODE_RANGE);
    lv_slider_set_value(slider, 70, LV_ANIM_OFF);
    lv_slider_set_left_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(e);

    /*Provide some extra space for the value*/
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        lv_coord_t * size = lv_event_get_param(e);
        *size = LV_MAX(*size, 50);
    }
}

```

(下页继续)

(续上页)

```

else if(code == LV_EVENT_DRAW_PART_END) {
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
    if(dsc->part == LV_PART_INDICATOR) {
        char buf[16];
        lv_snprintf(buf, sizeof(buf), "%d - %d", lv_slider_get_left_value(obj), ↵
                    lv_slider_get_value(obj));

        lv_point_t label_size;
        lv_txt_get_size(&label_size, buf, LV_FONT_DEFAULT, 0, 0, LV_COORD_MAX, 0);
        lv_area_t label_area;
        label_area.x1 = dsc->draw_area->x1 + lv_area_get_width(dsc->draw_area) / ↵
        2 - label_size.x / 2;
        label_area.x2 = label_area.x1 + label_size.x;
        label_area.y2 = dsc->draw_area->y1 - 10;
        label_area.y1 = label_area.y2 - label_size.y;

        lv_draw_label_dsc_t label_draw_dsc;
        lv_draw_label_dsc_init(&label_draw_dsc);

        lv_draw_label(&label_area, dsc->clip_area, &label_draw_dsc, buf, NULL);
    }
}
#endif

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/slider/lv_example_slider_3.py

2.7.23 Span

2.7.24 Spinbox

2.7.25 Spinner

2.7.26 Switch

2.7.27 Table

Simple table

```

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

static void draw_part_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
    /*If the cells are drawn...*/
    if(dsc->part == LV_PART_ITEMS) {
        uint32_t row = dsc->id / lv_table_get_col_cnt(obj);
        uint32_t col = dsc->id - row * lv_table_get_col_cnt(obj);
    }
}

```

(下页继续)

(续上页)

```

/*Make the texts in the first cell center aligned*/
if(row == 0) {
    dsc->label_dsc->align = LV_TEXT_ALIGN_CENTER;
    dsc->rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_BLUE), u
→dsc->rect_dsc->bg_color, LV_OPA_20);
    dsc->rect_dsc->bg_opa = LV_OPA_COVER;
}
/*In the first column align the texts to the right*/
else if(col == 0) {
    dsc->label_dsc->flag = LV_TEXT_ALIGN_RIGHT;
}

/*MAKE every 2nd row grayish*/
if((row != 0 && row % 2 == 0) {
    dsc->rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_GREY), u
→dsc->rect_dsc->bg_color, LV_OPA_10);
    dsc->rect_dsc->bg_opa = LV_OPA_COVER;
}
}

void lv_example_table_1(void)
{
    lv_obj_t * table = lv_table_create(lv_scr_act());

    /*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_part_event_cb, LV_EVENT_DRAW_PART_BEGIN, NULL);
}

#endif

```

```

# Create a normal cell style
style_cell1 = lv.style_t()
lv.style_copy(style_cell1, lv.style_plain)
style_cell1.body.border.width = 1
style_cell1.body.border.color = lv.color_make(0,0,0)

# Create a header cell style
style_cell2 = lv.style_t()
lv.style_copy(style_cell2, lv.style_plain)
style_cell2.body.border.width = 1
style_cell2.body.border.color = lv.color_make(0,0,0)
style_cell2.body.main_color = lv.color_make(0xC0, 0xC0, 0xC0)
style_cell2.body.grad_color = lv.color_make(0xC0, 0xC0, 0xC0)

table = lv.table(lv.scr_act())
table.set_style(lv.table.STYLE.CELL1, style_cell1)
table.set_style(lv.table.STYLE.CELL2, style_cell2)
table.set_style(lv.table.STYLE.BG, lv.style_transp_tight)
table.set_col_cnt(2)
table.set_row_cnt(4)
table.align(None, lv.ALIGN.CENTER, 0, 0)

# Make the cells of the first row center aligned
table.set_cell_align(0, 0, lv.label.ALIGN.CENTER)
table.set_cell_align(0, 1, lv.label.ALIGN.CENTER)

# Make the cells of the first row TYPE = 2 (use `style_cell2`)
table.set_cell_type(0, 0, 2)
table.set_cell_type(0, 1, 2)

# Fill the first column
table.set_cell_value(0, 0, "Name")
table.set_cell_value(1, 0, "Apple")
table.set_cell_value(2, 0, "Banana")
table.set_cell_value(3, 0, "Citron")

# Fill the second column
table.set_cell_value(0, 1, "Price")
table.set_cell_value(1, 1, "$7")
table.set_cell_value(2, 1, "$4")
table.set_cell_value(3, 1, "$6")

```

Lightweighted list from table

```

#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(e);
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
    /*If the cells are drawn...*/
    if(dsc->part == LV_PART_ITEMS) {

```

(下页继续)

(续上页)

```

bool chk = lv_table_has_cell_ctrl(obj, dsc->id, 0, LV_TABLE_CELL_CTRL_CUSTOM_1);

    lv_rect_dsc_t rect_dsc;
    lv_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 = dsc->draw_area->x2 - 50;
    sw_area.x2 = sw_area.x1 + 40;
    sw_area.y1 = dsc->draw_area->y1 + lv_area_get_height(dsc->draw_area) / 2 - 10;
    sw_area.y2 = sw_area.y1 + 20;
    lv_rect(&sw_area, dsc->clip_area, &rect_dsc);

    rect_dsc.bg_color = lv_color_white();
    if(chk) {
        sw_area.x2 -= 2;
        sw_area.x1 = sw_area.x2 - 16;
    } else {
        sw_area.x1 += 2;
        sw_area.x2 = sw_area.x1 + 16;
    }
    sw_area.y1 += 2;
    sw_area.y2 -= 2;
    lv_rect(&sw_area, dsc->clip_area, &rect_dsc);
}

static void change_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    uint16_t col;
    uint16_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_add_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_scr_act());
}

```

(下页继续)

(续上页)

```

/*Set a smaller height to the table. It'll make it scrollable*/
lv_obj_set_size(table, 150, 200);

lv_table_set_col_width(table, 0, 150);
lv_table_set_row_cnt(table, ITEM_CNT); /*Not required but avoids a lot of memory
→reallocation lv_table_set_set_value*/
lv_table_set_col_cnt(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 %d", i + 1);
}

lv_obj_align(table, LV_ALIGN_CENTER, 0, -20);

/*Add an event callback to apply some custom drawing*/
lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_PART_END, NULL);
lv_obj_add_event_cb(table, change_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

lv_mem_monitor_t mon2;
lv_mem_monitor(&mon2);

uint32_t mem_used = mon1.free_size - mon2.free_size;

uint32_t elaps = lv_tick_elaps(t);

lv_obj_t * label = lv_label_create(lv_scr_act());
lv_label_set_text_fmt(label, "%d items were created in %d ms\n"
                      "using %d bytes of memory",
                      ITEM_CNT, elaps, mem_used);

lv_obj_align(label, LV_ALIGN_BOTTOM_MID, 0, -10);

}

#endif

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/table/lv_example_table_2.py

2.7.28 Tabview

2.7.29 Textarea

Simple Text area

```

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

static void textarea_event_handler(lv_event_t * e)

```

(下页继续)

(续上页)

```

{
    lv_obj_t * ta = lv_event_get_target(e);
    LV_LOG_USER("Enter was pressed. The current text is: %s", lv_textarea_get_
    ↪text(ta));
}

static void btnm_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_t * ta = lv_event_get_user_data(e);
    const char * txt = lv_btnmatrix_get_btn_text(obj, lv_btnmatrix_get_selected_
    ↪btn(obj));

    if(strcmp(txt, LV_SYMBOL_BACKSPACE) == 0) lv_textarea_del_char(ta);
    else if(strcmp(txt, LV_SYMBOL_NEW_LINE) == 0) lv_textarea_add_char(ta, '\n');
    else lv_textarea_add_text(ta, txt);
}

void lv_example_textarea_1(void)
{
    lv_obj_t * ta = lv_textarea_create(lv_scr_act());
    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_btnmatrix_create(lv_scr_act());
    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_clear_flag(btnm, LV_OBJ_FLAG_CLICK_FOCUSABLE); /*To keep the text area
    ↪focused on button clicks*/
    lv_btnmatrix_set_map(btnm, btnm_map);
}

#endif

```

```

def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
        print("Value: %s" % obj.get_text())
    elif event == lv.EVENT.LONG_PRESSED_REPEAT:
        # For simple test: Long press the Text area to add the text below
        tal.add_text("\n\nYou can scroll it if the text is long enough.\n")

tal = lv.ta(lv.scr_act())
tal.set_size(200, 100)
tal.align(None, lv.ALIGN.CENTER, 0, 0)
tal.set_cursor_type(lv.CURSOR.BLOCK)
tal.set_text("A text in a Text Area")      # Set an initial text
tal.set_event_cb(event_handler)

```

Text area with password field

```
#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_scr_act());
    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_scr_act());
    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_scr_act());
    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_scr_act());
    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_scr_act());
    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*/
}

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(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) {

```

(下页继续)

(续上页)

```

const char * str = lv_event_get_param(e);
if(str[0] == '\n') {
    LV_LOG_USER("Ready\n");
}
}

#endif

```

```

HOR_RES = lv.disp_get_hor_res(lv.disp_get_default())

def kb_event_cb(event_kb, event):
    # Just call the regular event handler
    event_kb.def_event_cb(event)

def ta_event_cb(ta, event):
    if event == lv.EVENT.INSERT:
        # get inserted value
        ptr = lv.C_Pointer()
        ptr.ptr_val = lv.event_get_data()
        if ptr.str_val == "\n":
            print("Ready")
    elif event == lv.EVENT.CLICKED:
        # Focus on the clicked text area
        kb.set_ta(ta)

# Create the password box
pwd_ta = lv.ta(lv.scr_act())
pwd_ta.set_text("");
pwd_ta.set_pwd_mode(True)
pwd_ta.set_one_line(True)
pwd_ta.set_width(HOR_RES // 2 - 20)
pwd_ta.set_pos(5, 20)
pwd_ta.set_event_cb(ta_event_cb)

# Create a label and position it above the text box
pwd_label = lv.label(lv.scr_act())
pwd_label.set_text("Password:")
pwd_label.align(pwd_ta, lv.ALIGN.OUT_TOP_LEFT, 0, 0)

# Create the one-line mode text area
oneline_ta = lv.ta(lv.scr_act(), pwd_ta)
oneline_ta.set_pwd_mode(False)
oneline_ta.set_cursor_type(lv.CURSOR.LINE | lv.CURSOR.HIDDEN)
oneline_ta.align(None, lv.ALIGN.IN_TOP_RIGHT, -5, 20)
oneline_ta.set_event_cb(ta_event_cb)

# Create a label and position it above the text box
oneline_label = lv.label(lv.scr_act())
oneline_label.set_text("Text:")
oneline_label.align(oneline_ta, lv.ALIGN.OUT_TOP_LEFT, 0, 0)

# Create a keyboard and make it fill the width of the above text areas
kb = lv.kb(lv.scr_act())
kb.set_pos(5, 90)
kb.set_event_cb(kb_event_cb) # Setting a custom event handler stops the keyboard from
                           // closing automatically

```

(下页继续)

(续上页)

```
kb.set_size(HOR_RES - 10, 140)

kb.set_ta(pwd_ta) # Focus it on one of the text areas to start
kb.set_cursor_manage(True) # Automatically show/hide cursors on text areas
```

Text auto-formatting

```
#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_scr_act());
    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_scr_act());
    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(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
```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/textarea/lv_example_textarea_3.py

2.7.30 Tabview

2.7.31 Window

CHAPTER 3

Get started (开始)

There are several ways to get your feet wet with LVGL. Here is one recommended order of documents to read and things to play with when you are learning to use LVGL:

有几种方法可以让您快速上手 LVGL。以下是您在学习使用 LVGL 时推荐阅读的文档顺序和玩法：

1. Check the [Online demos](#) to see LVGL in action (3 minutes)
2. Read the [Introduction](#) page of the documentation (5 minutes)
3. Read the [Quick overview](#) page of the documentation (15 minutes)
4. Set up a [Simulator](#) (10 minutes)
5. Try out some [Examples](#)
6. Port LVGL to a board. See the [Porting](#) guide or check the ready to use [Projects](#)
7. Read the [Overview](#) page to get a better understanding of the library. (2-3 hours)
8. Check the documentation of the [Widgets](#) to see their features and usage
9. If you have questions got to the [Forum](#)
10. Read the [Contributing](#) guide to see how you can help to improve LVGL (15 minutes)
 1. 查看一些[在线演示](#)来了解 LVGL 有哪些实际应用 (3 分钟)
 2. 阅读文档的[介绍](#)页面 (5 分钟)
 3. 阅读文档的[快速概览](#)页面 (15 分钟)
 4. 设置[模拟器](#) (10 分钟)
 5. 尝试一些[示例](#)
 6. 移植 LVGL 到你的开发板上。请参阅[移植](#)指南或在仓库中检查你准备使用的[项目](#)
 7. 阅读[概述](#)页面以更好地了解库。(2-3 小时)
 8. 查看[Widgets](#) 的文档以查看它们的功能和用法
 9. 如有问题请到[论坛](#)发帖提问

- 阅读 [贡献](#) 指南，了解如何帮助改进 LVGL (15 分钟)

3.1 Quick overview (快速概览)

Here you can learn the most important things about LVGL. You should read this first to get a general impression and read the detailed [Porting](#) and [Overview](#) sections after that.

在这里您可以了解有关 LVGL 的最重要的事情。您应该先阅读本文以获得大致印象，然后再阅读详细的[移植](#)和[概述](#)部分。

3.1.1 Get started in a simulator (从模拟器开始)

Instead of porting LVGL to embedded hardware straight away, it's highly recommended to get started in a simulator first. 强烈建议您先在 lvgl 模拟器上开始学习实验，而不是立即将 LVGL 移植到嵌入式硬件。

LVGL is ported to many IDEs to be sure you will find your favorite one. Go to the [Simulators](#) section to get ready-to-use projects that can be run on your PC. This way you can save the time of porting for now and get some experience with LVGL immediately.

LVGL 已适配到许多 IDE，以确保您能找到自己喜欢的一种模拟器开发环境。转到[模拟器](#)部分以获取可以在您的 PC 上运行的即用型项目。通过这种方式，您可以暂时节省移植时间并立即获得一些使用 LVGL 的经验。(这是非常有用的！)

3.1.2 Add LVGL into your project (将 LVGL 添加到您的项目中)

If you would rather try LVGL on your own project follow these steps:

如果您更愿意在自己的项目中尝试 LVGL，请按照以下步骤操作：

- [Download](#) or clone the library from GitHub with `git clone https://github.com/lvgl/lvgl.git`.
- Copy the `lvgl` folder into your project.
- Copy `lvgl/lv_conf_template.h` as `lv_conf.h` next to the `lvgl` folder, change the first `#if 0` to `1` to enable the file's content and set the `LV_COLOR_DEPTH` defines.
- Include `lvgl/lvgl.h` in files where you need to use LVGL related functions.
- Call `lv_tick_inc(x)` every `x` milliseconds in a Timer or Task (`x` should be between 1 and 10). It is required for the internal timing of LVGL. Alternatively, configure `LV_TICK_CUSTOM` (see `lv_conf.h`) so that LVGL can retrieve the current time directly.
- Call `lv_init()`
- Create a draw buffer: LVGL will render the graphics here first, and send the rendered image to the display. The buffer size can be set freely but 1/10 screen size is a good starting point.
- 使用 `git` 命令 `git clone https://github.com/lvgl/lvgl.git` 从 GitHub 下载或克隆库。
- 将 `lvgl` 文件夹复制到您的项目中。
- 将 `lvgl/lv_conf_template.h` 作为 `lv_conf.h` 复制到 `lvgl` 文件夹旁边，将其第一个的 `#if 0` 更改为 `1` 以便能文件的内容并修改设置 `LV_COLOR_DEPTH` 宏。
- 在需要使用 LVGL 相关函数的文件中包含 `lvgl/lvgl.h`。
- 在计时器或任务中每 `x` 毫秒调用一次 `lv_tick_inc(x)` (`x` 应该在 1 到 10 之间)。LVGL 的内部时序需要它。或者，配置 `LV_TICK_CUSTOM` (参见 `lv_conf.h`)，以便 LVGL 可以直接检索当前时间。

- 调用 `lv_init()` (初始化 lvgl 库)
- 创建一个绘制缓冲区：LVGL 将首先在此处渲染图形，并将渲染的图像发送到显示器。缓冲区大小可以自由设置，但 1/10 屏幕大小是一个很好的起点。

```
static lv_disp_draw_buf_t draw_buf;
static lv_color_t buf1[DISP_HOR_RES * DISP_VER_RES / 10];
/*Declare a buffer for 1/10 screen size*/
lv_disp_draw_buf_init(&draw_buf, buf1, NULL, MY_DISP_HOR_RES * MY_DISP_VER_SER / 10);
/*Initialize the display buffer.*
```

- Implement and register a function which can copy the rendered image to an area of your display:

实现并注册一个函数，该函数可以将渲染图像复制到显示区域：

```
lv_disp_drv_t disp_drv;           /*Descriptor of a display driver*/
lv_disp_drv_init(&disp_drv);      /*Basic initialization*/
disp_drv.flush_cb = my_disp_flush; /*Set your driver function*/
disp_drv.buffer = &draw_buf;       /*Assign the buffer to the display*/
disp_drv.hor_res = MY_DISP_HOR_RES; /*Set the horizontal resolution of the display*/
disp_drv.ver_res = MY_DISP_VER_RES; /*Set the vertical resolution of the display*/
lv_disp_drv_register(&disp_drv);   /*Finally register the driver*/

void my_disp_flush(lv_disp_drv_t * disp, const lv_area_t * area, lv_color_t * color_p)
{
    int32_t x, y;
    /*It's a very slow but simple implementation.
     *`set_pixel` needs to be written by you to a set pixel on the screen*/
    for(y = area->y1; y <= area->y2; y++) {
        for(x = area->x1; x <= area->x2; x++) {
            set_pixel(x, y, *color_p);
            color_p++;
        }
    }
    lv_disp_flush_ready(disp);      /* Indicate you are ready with the flushing*/
}
```

- Implement and register a function which can read an input device. E.g. for a touch pad:

• 实现并注册一个可以读取输入设备的函数。例如。对于触摸板：

```
lv_indev_drv_t indev_drv;           /*Descriptor of a input device driver*/
lv_indev_drv_init(&indev_drv);      /*Basic initialization*/
indev_drv.type = LV_INDEV_TYPE_POINTER; /*Touch pad is a pointer-like device*/
indev_drv.read_cb = my_touchpad_read; /*Set your driver function*/
lv_indev_drv_register(&indev_drv);   /*Finally register the driver*/

bool my_touchpad_read(lv_indev_t * indev, lv_indev_data_t * data)
{
    /*`touchpad_is_pressed` and `touchpad_get_xy` needs to be implemented by you*/
    if(touchpad_is_pressed()) {
        data->state = LV_INDEV_STATE_PRESSED;
        touchpad_get_xy(&data->point.x, &data->point.y);
    } else {
        data->state = LV_INDEV_STATE_RELEASED;
    }
}
```

- Call `lv_timer_handler()` periodically every few milliseconds in the main `while(1)` loop or in an operating system task. It will redraw the screen if required, handle input devices, animation etc.
- 在主 `while(1)` 循环或操作系统任务中每隔几毫秒定期调用 `lv_timer_handler()`。如果需要，它将重绘屏幕，处理输入设备，动画等。

For a more detailed guide go to the [Porting](#) section.

有关更详细的指南，请转到[移植](#)部分。

3.1.3 Learn the basics (学习基础知识)

Widgets (部件)

The graphical elements like Buttons, Labels, Sliders, Charts etc. are called objects or widgets. Go to [Widgets](#) to see the full list of available widgets.

按钮、标签、滑块、图表等图形元素称为对象或小部件。转到[部件](#)以查看可用小部件的完整列表。

Every object has a parent object where it is created. For example if a label is created on a button, the button is the parent of label.

每个对象都有一个创建它的父对象。例如，如果在按钮上创建标签，则该按钮是标签的父级。

The child object moves with the parent and if the parent is deleted the children will be deleted too.

子对象与父对象一起移动，如果删除父对象，子对象也将被删除。

Children can be visible only on their parent. In other words, the parts of the children outside of the parent are clipped.

子项只能在其父项上可见。换句话说，父级之外的子级部分被剪掉了。

A Screen is the "root" parent. You can have any number of screens.

Screen 是“根”父级。您可以拥有任意数量的屏幕。

To get the current screen call `lv_scr_act()`, and to load a screen use `lv_scr_load(scr1)`.

要获取当前屏幕调用 `lv_scr_act()`，并使用 `lv_scr_load(scr1)` 加载屏幕。

You can create a new object with `lv_<type>_create(parent)`. It will return an `lv_obj_t *` variable that can be used as a reference to the object to set its parameters.

您可以使用 `lv_<type>_create(parent)` 创建一个新对象。它将返回一个 `lv_obj_t *` 变量，该变量可用作对象的引用以设置其参数。

For example (例如) :

```
lv_obj_t * slider1 = lv_slider_create(lv_scr_act());
```

To set some basic attributes `lv_obj_set_<parameter_name>(obj, <value>)` functions can be used. For example:

要设置一些基本属性，可以使用 `lv_obj_set_<parameter_name>(obj, <value>)` 函数。例如：

```
lv_obj_set_x(btn1, 30);
lv_obj_set_y(btn1, 10);
lv_obj_set_size(btn1, 200, 50);
```

The widgets have type specific parameters too which can be set by `lv_<widget_type>_set_<parameter_name>(obj, <value>)` functions. For example:

这些小部件也具有类型特定的参数，可以通过 `lv_<widget_type>_set_<parameter_name>(obj, <value>)` 函数设置。例如：

```
lv_slider_set_value(slider1, 70, LV_ANIM_ON);
```

To see the full API visit the documentation of the widgets or the related header file (e.g. [lvgl/src/widgets/lv_slider.h](#)).

要查看完整的 API，请访问小部件的文档或相关的头文件（例如 [lvgl/src/widgets/lv_slider.h](#)）。

Events (事件)

Events are used to inform the user that something has happened with an object. You can assign one or more callbacks to an object which will be called if the object is clicked, released, dragged, being deleted etc.

A callback is assigned like this:

事件用于通知用户某个对象发生了某些事情。您可以将一个或多个回调分配给一个对象，如果该对象被单击、释放、拖动、删除等将被调用。

一个回调是这样分配的：

```
lv_obj_add_event_cb(btn, btn_event_cb, LV_EVENT_CLICKED, NULL); /*Assign a callback
to the button*/
...
void btn_event_cb(lv_event_t * e)
{
    printf("Clicked\n");
}
```

Instead of `LV_EVENT_CLICKED` `LV_EVENT_ALL` can be used too to call the callback for any event.

From `lv_event_t * e` the current event code can be get with

代替 `LV_EVENT_CLICKED`、`LV_EVENT_ALL` 也可用于调用任何事件的回调。

从 `lv_event_t * e` 可以得到当前的事件代码

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

The object that triggered the event can be retrieved with

触发事件的对象可以用

```
lv_obj_t * obj = lv_event_get_target(e);
```

To learn all features of the events go to the [Event overview](#) section.

要了解事件的所有功能，请转到事件概述 部分。

Parts (部分)

Widgets might be built from one or more *parts*. For example a button has only one part called `LV_PART_MAIN`. However, a *Slider* has `LV_PART_MAIN`, `LV_PART_INDICATOR` and `LV_PART_KNOB`.

部件可能由一个或多个部分构建。例如，一个按钮只有一个名为 `LV_PART_MAIN` 的部分。但是，滑块具有 `LV_PART_MAIN`、`LV_PART_INDICATOR` 和 `LV_PART_KNOB`。

By using parts you can apply different styles to different parts. (See below)

通过使用零件，您可以将不同的样式应用于不同的零件。(见下文)

To learn which parts are used by which object read the widgets' documentation.

要了解哪个对象使用了哪些部件，请阅读部件的文档。

States (状态)

The objects can be in a combination of the following states:

对象可以处于以下状态的组合：

- `LV_STATE_DEFAULT` Normal, released state
- `LV_STATE_CHECKED` Toggled or checked state
- `LV_STATE_FOCUSED` Focused via keypad or encoder or clicked via touchpad/mouse
- `LV_STATE_FOCUS_KEY` Focused via keypad or encoder but not via touchpad/mouse
- `LV_STATE_EDITED` Edit by an encoder
- `LV_STATE_HOVERED` Hovered by mouse (not supported now)
- `LV_STATE_PRESSED` Being pressed
- `LV_STATE_SCROLLED` Being scrolled
- `LV_STATE_DISABLED` Disabled
- `LV_STATE_DEFAULT` 正常，释放状态
- `LV_STATE_CHECKED` 切换或选中状态
- `LV_STATE_FOCUSED` 通过键盘或编码器聚焦或通过触摸板/鼠标点击
- `LV_STATE_FOCUS_KEY` 通过键盘或编码器聚焦，但不通过触摸板/鼠标聚焦
- `LV_STATE_EDITED` 由编码器编辑
- `LV_STATE_HOVERED` 鼠标悬停（现在不支持）
- `LV_STATE_PRESSED` 被按下
- `LV_STATE_SCROLLED` 正在滚动
- `LV_STATE_DISABLED` 禁用

For example, if you press an object it will automatically go to `LV_STATE_FOCUSED` and `LV_STATE_PRESSED` state and when you release it, the `LV_STATE_PRESSED` state will be removed.

例如，如果你按下一个对象，它会自动进入 `LV_STATE_FOCUSED` 和 `LV_STATE_PRESSED` 状态，当你释放它时，`LV_STATE_PRESSED` 状态将被移除。

To check if an object is in a given state use `lv_obj_has_state(obj, LV_STATE_...)`. It will return `true` if the object is in that state at that time.

To manually add or remove states use

要检查对象是否处于给定状态, 请使用 `lv_obj_has_state(obj, LV_STATE_...)`。如果对象当时处于该状态, 它将返回 `true`。

要手动添加或删除状态, 请使用下面的函数

```
lv_obj_add_state(obj, LV_STATE_...);
lv_obj_clear_state(obj, LV_STATE_...);
```

Styles (样式)

Styles contains properties such as background color, border width, font, etc to describe the appearance of the objects.

样式包含诸如背景颜色、边框宽度、字体等属性来描述对象的外观。

The styles are `lv_style_t` variables. Only their pointer is saved in the objects so they need to be static or global.

样式是 `lv_style_t` 变量。只有它们的指针保存在对象中, 因此它们需要是静态的或全局的。

Before using a style it needs to be initialized with `lv_style_init(&style1)`. After that properties can be added. For example:

在使用样式之前, 它需要使用 `lv_style_init(&style1)` 进行初始化。之后可以添加属性。例如:

```
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))
```

See the full list of properties [here](#).

在这里 查看完整的属性列表。

The styles are assigned to an object's part and state. For example to "Use this style on the slider's indicator when the slider is pressed":

样式被分配给对象的部分和状态。例如 “按下滑块时在滑块指示器上使用此样式”:

```
lv_obj_add_style(slider1, &style1, LV_PART_INDICATOR | LV_STATE_PRESSED);
```

If the `part` is `LV_PART_MAIN` it can be omitted:

如果 `part` 是 `LV_PART_MAIN` 可以省略:

```
lv_obj_add_style(btn1, &style1, LV_STATE_PRESSED); /*Equal to LV_PART_MAIN | LV_STATE_PRESSED*/
```

Similarly, `LV_STATE_DEFAULT` can be omitted too:

类似地, `LV_STATE_DEFAULT` 也可以省略:

```
lv_obj_add_style(slider1, &style1, LV_PART_INDICATOR); /*Equal to LV_PART_INDICATOR | LV_STATE_DEFAULT*/
```

For `LV_STATE_DEFAULT` and `LV_PART_MAIN` simply write `0`:

对于 `LV_STATE_DEFAULT` 和 `LV_PART_MAIN` 只需写下 `0`:

```
lv_obj_add_style(btn1, &style1, 0); /*Equal to LV_PART_MAIN | LV_STATE_DEFAULT*/
```

The styles can be cascaded (similarly to CSS). It means you can add more styles to a part of an object. For example `style_btn` can set a default button appearance, and `style_btn_red` can overwrite the background color to make the button red:

样式可以级联（类似于 CSS）。这意味着您可以为对象的一部分添加更多样式。例如 `style_btn` 可以设置默认按钮外观，`style_btn_red` 可以覆盖背景颜色使按钮变为红色：

```
lv_obj_add_style(btn1, &style_btn, 0);
lv_obj_add_style(btn1, &style1_btn_red, 0);
```

If a property is not set on for the current state the style with `LV_STATE_DEFAULT` will be used. If the property is not defined even in the default state a default value is used.

如果没有为当前状态设置属性，则将使用带有“`LV_STATE_DEFAULT`”的样式。如果即使在默认状态下也未定义该属性，则使用默认值。

Some properties (typically the text-related ones) can be inherited. It means if a property is not set in an object it will be searched in its parents too. For example, you can set the font once in the screen's style and all text on that screen will inherit it by default.

一些属性（通常是与文本相关的）可以被继承。这意味着如果一个属性没有在一个对象中设置，它也会在它的父级中搜索。例如，您可以在屏幕样式中设置一次字体，该屏幕上的所有文本都会默认继承它。

Local style properties also can be added to the objects. It creates a style which resides inside the object and which is used only by the object:

本地样式属性也可以添加到对象中。它创建了一个位于对象内部并且仅由对象使用的样式：

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

To learn all the features of styles see the [Style overview](#) section.

要了解样式的功能，请参阅[样式概述](#)部分。

Themes

Themes are the default styles of the objects. The styles from the themes are applied automatically when the objects are created.

You can select the theme to use in `lv_conf.h`.

主题是对象的默认样式。创建对象时，将自动应用来自主题的样式。

您可以在 `lv_conf.h` 中选择要使用的主题。

3.1.4 Examples

3.1.5 Micropython

Learn more about [Micropython](#).

了解有关[Micropython](#) 的更多信息。

```
# Create a Button and a Label
scr = lv.obj()
btn = lv.btn(scr)
btn.align(lv.scr_act(), lv.ALIGN.CENTER, 0, 0)
```

(下页继续)

(续上页)

```

label = lv.label(btn)
label.set_text("Button")

# Load the screen
lv.scr_load(scr)

```

3.2 Simulator on PC (PC 上的模拟器)

You can try out LVGL **using only your PC** (i.e. without any development boards). LVGL will run on a simulator environment on the PC where anyone can write and experiment the real LVGL applications.

Using the simulator on the PC has the following advantages:

- Hardware independent - Write code, run it on the PC and see the result on the PC monitor.
- Cross-platform - Any Windows, Linux or MacOS system can run the PC simulator.
- Portability - the written code is portable, which means you can simply copy it when using an embedded hardware.
- Easy Validation - The simulator is also very useful to report bugs because it means common platform for every user. So it's a good idea to reproduce a bug in the simulator and use the code snippet in the [Forum](#).

您可以 **仅使用您的 PC** (即没有任何开发板) 来试用 LVGL。LVGL 将在 PC 上的模拟器环境中运行，任何人都可以在其中编写和试验真正的 LVGL 应用程序。

在 PC 上使用模拟器运行 lvgl 有以下优点：

- 独立于硬件 - 编写代码，在 PC 上运行它并在 PC 显示器上查看结果。
- 跨平台 - 任何 Windows、Linux 或 MacOS 系统都可以运行 PC 模拟器。
- 可移植性——编写的代码是可移植的，这意味着您可以在使用嵌入式硬件时简单地复制它。
- Easy Validation - 模拟器对于报告错误也非常有用，因为它意味着每个用户的通用平台。所以最好在模拟器中重现一个错误并使用 [论坛](#) 中的代码片段。

3.2.1 Select an IDE (选择适合的 IDE)

The simulator is ported to various IDEs (Integrated Development Environments). Choose your favorite IDE, read its README on GitHub, download the project, and load it to the IDE.

- [Eclipse with SDL driver](#): Recommended on Linux and Mac
- [CodeBlocks](#): Recommended on Windows (简单方便推荐使用)
- [VisualStudio with SDL driver](#): For Windows
- [VSCode with SDL driver](#): Recommended on Linux and Mac
- [PlatformIO with SDL driver](#): Recommended on Linux and Mac

模拟器被移植到各种 IDE (集成开发环境)。选择您最喜欢的 IDE，在 GitHub 上阅读其 README，下载项目，然后将其加载到 IDE。

- [Eclipse with SDL driver](#): Linux 和 Mac
- [CodeBlocks](#): Windows
- [VisualStudio with SDL driver](#): Windows

- VSCode with SDL driver: Linux 和 Mac
- PlatformIO with SDL driver: Linux 和 Mac

You can use any IDE for the development but, for simplicity, the configuration for Eclipse CDT is what we'll focus on in this tutorial. The following section describes the set-up guide of Eclipse CDT in more details.

Note: If you are on Windows, it's usually better to use the Visual Studio or CodeBlocks projects instead. They work out of the box without requiring extra steps.

您可以使用任何 IDE 进行开发，但为简单起见，Eclipse CDT 的配置是我们在本教程中重点关注的内容。以下部分更详细地描述了 Eclipse CDT 的设置指南。

注意：如果您使用的是 Windows，通常最好改用 Visual Studio 或 CodeBlocks 项目。它们开箱即用，无需额外步骤。

3.2.2 Set-up Eclipse CDT (使用 Eclipse CDT 开发)

Install Eclipse CDT (安装 Eclipse CDT)

Eclipse CDT is a C/C++ IDE.

Eclipse is a Java based software therefore be sure **Java Runtime Environment** is installed on your system.

On Debian-based distros (e.g. Ubuntu): `sudo apt-get install default-jre`

Note: If you are using other distros, then please refer and install 'Java Runtime Environment' suitable to your distro. Note: If you are using macOS and get a "Failed to create the Java Virtual Machine" error, uninstall any other Java JDK installs and install Java JDK 8u. This should fix the problem.

You can download Eclipse's CDT from: <https://www.eclipse.org/cdt/downloads.php>. Start the installer and choose *Eclipse CDT* from the list.

Eclipse 是基于 Java 的软件，因此请确保您的系统上安装了 **Java 运行时环境**。

在基于 Debian 的发行版（例如 Ubuntu）上: `sudo apt-get install default-jre`

注意：如果您使用其他发行版，请参考并安装适合您的发行版的“Java 运行时环境”。注意：如果您使用的是 macOS 并收到“无法创建 Java 虚拟机”错误，请卸载任何其他 Java JDK 安装并安装 Java JDK 8u。这应该可以解决问题。

您可以从以下位置下载 Eclipse 的 CDT: <https://www.eclipse.org/cdt/downloads.php>。启动安装程序并从列表中选择 Eclipse CDT。

Install SDL 2 (安装 SDL 2)

The PC simulator uses the **SDL 2** cross platform library to simulate a TFT display and a touch pad.

PC 模拟器使用 **SDL 2** 跨平台库来模拟 TFT 显示器和触摸板。

Linux

On **Linux** you can easily install SDL2 using a terminal:

1. Find the current version of SDL2: `apt-cache search libsdl2` (e.g. `libsdl2-2.0-0`)
2. Install SDL2: `sudo apt-get install libsdl2-2.0-0` (replace with the found version)
3. Install SDL2 development package: `sudo apt-get install libsdl2-dev`
4. If build essentials are not installed yet: `sudo apt-get install build-essential`

在 **Linux** 上，您可以使用终端轻松安装 SDL2：

1. 找到 SDL2 的当前版本: `apt-cache search libsdl2` (e.g. `libsdl2-2.0-0`)
2. 安装 SDL2: `sudo apt-get install libsdl2-2.0-0` (替换为找到的版本)
3. 安装 SDL2 开发包: `sudo apt-get install libsdl2-dev`
4. 如果尚未安装 build Essentials: `sudo apt-get install build-essential`

Windows

If you are using **Windows** firstly you need to install MinGW (64 bit version). After installing MinGW, do the following steps to add SDL2:

1. Download the development libraries of SDL. Go to <https://www.libsdl.org/download-2.0.php> and download *Development Libraries: SDL2-devel-2.0.5-mingw.tar.gz*
2. Decompress the file and go to `x86_64-w64-mingw32` directory (for 64 bit MinGW) or to `i686-w64-mingw32` (for 32 bit MinGW)
3. Copy `...mingw32/include/SDL2` folder to `C:/MinGW/.../x86_64-w64-mingw32/include`
4. Copy `...mingw32/lib/` content to `C:/MinGW/.../x86_64-w64-mingw32/lib`
5. Copy `...mingw32/bin/SDL2.dll` to `{eclipse_workapce}/pc_simulator/Debug/`. Do it later when Eclipse is installed.

Note: If you are using **Microsoft Visual Studio** instead of Eclipse then you don't have to install MinGW.

如果您使用的是 Windows，则需要安装 MinGW（64 位版本）。安装 MinGW 后，执行以下步骤添加 SDL2：

1. 下载 SDL 的开发库。打开 <https://www.libsdl.org/download-2.0.php> 并下载开发库: `SDL2-devel-2.0.5-mingw.tar.gz`
2. 解压文件并进入 `x86_64-w64-mingw32` 目录（对于 64 位 MinGW）或 `i686-w64-mingw32`（对于 32 位 MinGW）
3. 将 `...mingw32/include/SDL2` 文件夹复制到 `C:/MinGW/.../x86_64-w64-mingw32/include`
4. 将 `...mingw32/lib/` 内容复制到 `C:/MinGW/.../x86_64-w64-mingw32/lib`
5. 将 `...mingw32/bin/SDL2.dll` 复制到 `{eclipse_workapce}/pc_simulator/Debug/`。稍后在安装 Eclipse 时执行此操作。

注意：如果您使用 **Microsoft Visual Studio** 而不是 Eclipse，那么您不必安装 MinGW。

OSX

On **OSX** you can easily install SDL2 with brew: `brew install sdl2`

If something is not working, then please refer this tutorial to get started with SDL.

在 **OSX** 上，您可以使用 brew 轻松安装 SDL2: `brew install sdl2`

如果出现问题，请参阅 [这个教程](#) 以开始使用 SDL。

Pre-configured project (预配置项目)

A pre-configured graphics library project (based on the latest release) is always available to get started easily. You can find the latest one on [GitHub](#). (Please note that, the project is configured for Eclipse CDT).

预配置的图形库项目（基于最新版本）始终可以轻松上手。你可以在 [GitHub 仓库](#) 上找到最新的版本。（请注意，该项目是为 Eclipse CDT 配置的）。

Add the pre-configured project to Eclipse CDT (将预先配置的项目添加到 Eclipse CDT)

Run Eclipse CDT. It will show a dialogue about the **workspace path**. Before accepting the path, check that path and copy (and unzip) the downloaded pre-configured project there. After that, you can accept the workspace path. Of course you can modify this path but, in that case copy the project to the corresponding location.

Close the start up window and go to **File->Import** and choose **General->Existing project into Workspace**. **Browse the root directory** of the project and click **Finish**

On **Windows** you have to do two additional things:

- Copy the **SDL2.dll** into the project's Debug folder
- Right click on the project -> Project properties -> C/C++ Build -> Settings -> Libraries -> Add ... and add *mingw32* above *SDLmain* and *SDL*. (The order is important: *mingw32*, *SDLmain*, *SDL*)

运行 Eclipse CDT。它将显示有关 **工作区路径** 的对话。在接受路径之前，检查该路径并在那里复制（并解压缩）下载的预配置项目。之后，您可以接受工作区路径。当然，您可以修改此路径，但在这种情况下，将项目复制到相应位置。

关闭启动窗口并转到 **File->Import** 并选择 **General->Existing project into Workspace**。浏览项目根目录，**点击完成**

在 **Windows** 上，您必须做另外两件事：

- 将 **SDL2.dll** 复制到项目的 Debug 文件夹中
- 右键单击 项目-> 项目属性-> C/C++ 构建-> 设置-> 库-> 添加... 并在 *SDLmain* 和 *SDL* 上方添加 *_mingw32_*。(顺序很重要: *mingw32*、*SDLmain*、*SDL*)

Compile and Run (编译并运行)

Now you are ready to run LVGL on your PC. Click on the Hammer Icon on the top menu bar to Build the project. If you have done everything right, then you will not get any errors. Note that on some systems additional steps might be required to "see" SDL 2 from Eclipse but, in most of cases the configurations in the downloaded project is enough.

After a success build, click on the Play button on the top menu bar to run the project. Now a window should appear in the middle of your screen.

Now you are ready to use LVGL and begin development on your PC.

现在您已准备好在您的 PC 上运行 LVGL。单击顶部菜单栏上的锤子图标以构建项目。如果你做的一切都是正确的，那么你就不会出现任何错误。请注意，在某些系统上，从 Eclipse 中“查看”SDL 2 可能需要额外的步骤，但在大多数情况下，下载项目中的配置就足够了。

成功构建后，单击顶部菜单栏上的“播放”按钮以运行项目。现在，屏幕中间应该会出现一个窗口。

现在您已准备好使用 LVGL 并可以开始在您的 PC 上进行开发了！

3.3 STM32

TODO

3.4 NXP

NXP has integrated LVGL into the MCUXpresso SDK packages for several of their general purpose and crossover microcontrollers, allowing easy evaluation and migration into your product design. [Download an SDK for a supported board](#) today and get started with your next GUI application.

恩智浦 (NXP) 已将 LVGL 集成到 MCUXpresso SDK 包中，用于其多个通用和跨界微控制器，从而可以轻松评估和迁移到您的产品设计中。[立即下载支持的开发板的 SDK](#)，并开始使用您的下一个 GUI 应用程序。

3.4.1 Creating new project with LVGL (使用 LVGL 创建新项目)

Downloading the MCU SDK example project is recommended as a starting point. It comes fully configured with LVGL (and with PXP support if module is present), no additional integration work is required.

建议首先下载 MCU SDK 示例项目。它完全配置了 LVGL（如果存在模块，还支持 PXP），不需要额外的集成工作。

3.4.2 Adding HW acceleration for NXP iMX RT platforms using PXP (PiXel Pipeline) engine for existing projects

Several drawing features in LVGL can be offloaded to PXP engine. In order to use CPU time while PXP is running, RTOS is required to block the LVGL drawing thread and switch to another task, or simply to idle task, where CPU could be suspended to save power.

LVGL 中的几个绘图功能可以卸载到 PXP 引擎。为了在 PXP 运行时使用 CPU 时间，RTOS 需要阻塞 LVGL 绘图线程并切换到另一个任务，或者简单地切换到空闲任务，在那里 CPU 可以暂停以节省电量。

Features supported:

- RGB565 color format
- Area fill + optional transparency
- BLIT (BLock Image Transfer) + optional transparency
- Color keying + optional transparency
- Recoloring (color tint) + optional transparency
- RTOS integration layer
- Default FreeRTOS and bare metal code provided
- RGB565 颜色格式
- 区域填充 + 可选透明度
- BLIT (块图像传输) + 可选的透明度
- 颜色键控 + 可选透明度
- 重新着色 (色调) + 可选的透明度
- RTOS 集成层
- 提供默认的 FreeRTOS 和裸机代码

Basic configuration:

- Select NXP PXP engine in lv_conf.h: Set `LV_USE_GPU_NXP_PXP` to 1
- Enable default implementation for interrupt handling, PXP start function and automatic initialization: Set `LV_USE_GPU_NXP_PXP_AUTO_INIT` to 1
- If `FSL_RTOS_FREE_RTOS` symbol is defined, FreeRTOS implementation will be used, otherwise bare metal code will be included
- 在 lv_conf.h 中选择 NXP PXP 引擎：将 `LV_USE_GPU_NXP_PXP` 设置为 1
- 启用中断处理、PXP 启动功能和自动初始化的默认实现：将 `LV_USE_GPU_NXP_PXP_AUTO_INIT` 设置为 1
- 如果定义了 `FSL_RTOS_FREE_RTOS` 符号，将使用 FreeRTOS 实现，否则将包含裸机代码

Basic initialization:

- If `LV_USE_GPU_NXP_PXP_AUTO_INIT` is enabled, no user code is required; PXP is initialized automatically in `lv_init()`
- For manual PXP initialization, default configuration structure for callbacks can be used. Initialize PXP before calling `lv_init()`
- 如果启用 `LV_USE_GPU_NXP_PXP_AUTO_INIT`，则不需要用户代码；PXP 在 `lv_init()` 中自动初始化
- 对于手动 PXP 初始化，可以使用回调的默认配置结构。在调用 `lv_init()` 之前初始化 PXP

```

#ifndef LV_USE_GPU_NXP_PXP
    #include "lv_gpu/lv_gpu_nxp_pxp.h"
    #include "lv_gpu/lv_gpu_nxp_pxp_osa.h"
#endif

#ifndef LV_USE_GPU_NXP_PXP
    if (lv_gpu_nxp_pxp_init(&pxp_default_cfg) != LV_RES_OK) {
        PRINTF("PXP init error. STOP.\n");
        for ( ; ; );
    }
#endif

```

Project setup:

- Add PXP related files to project:
 - lv_gpu/lv_gpu_nxp.c, lv_gpu/lv_gpu_nxp.h: low level drawing calls for LVGL
 - lv_gpu/lv_gpu_nxp_osa.c, lv_gpu/lv_gpu_osa.h: default implementation of OS-specific functions (bare metal and FreeRTOS only)
 - * optional, required only if `LV_USE_GPU_NXP_PXP_AUTO_INIT` is set to 1
- PXP related code depends on two drivers provided by MCU SDK. These drivers need to be added to project:
 - fsl_pxp.c, fsl_pxp.h: PXP driver
 - fsl_cache.c, fsl_cache.h: CPU cache handling functions
- 将 PXP 相关文件添加到项目中:
 - lv_gpu/lv_gpu_nxp.c, lv_gpu/lv_gpu_nxp.h: LVGL 的低级绘图调用
 - lv_gpu/lv_gpu_nxp_osa.c、lv_gpu/lv_gpu_osa.h: 操作系统特定功能的默认实现（仅限裸机和 FreeRTOS）
 - * 可选, 仅当 `LV_USE_GPU_NXP_PXP_AUTO_INIT` 设置为 1 时才需要
 - PXP 相关代码依赖于 MCU SDK 提供的两个驱动程序。这些驱动程序需要添加到项目中:
 - * fsl_pxp.c、fsl_pxp.h: PXP 驱动程序
 - * fsl_cache.c、fsl_cache.h: CPU 缓存处理函数

Advanced configuration:

- Implementation depends on multiple OS-specific functions. Structure `lv_nxp_pxp_cfg_t` with callback pointers is used as a parameter for `lv_gpu_nxp_pxp_init()` function. Default implementation for FreeRTOS and baremetal is provided in `lv_gpu_nxp_osa.c`
 - `pxp_interrupt_init()`: Initialize PXP interrupt (HW setup, OS setup)
 - `pxp_interrupt_deinit()`: Deinitialize PXP interrupt (HW setup, OS setup)
 - `pxp_run()`: Start PXP job. Use OS-specific mechanism to block drawing thread. PXP must finish drawing before leaving this function.
- 实现取决于多个特定于操作系统的功能。带有回调指针的结构 `lv_nxp_pxp_cfg_t` 用作 `lv_gpu_nxp_pxp_init()` 函数的参数。`lv_gpu_nxp_osa.c` 中提供了 FreeRTOS 和裸机的默认实现
 - `pxp_interrupt_init()`: 初始化 PXP 中断（硬件设置，操作系统设置）

- `pxp_interrupt_deinit()`: 取消初始化 PXP 中断（硬件设置，操作系统设置）
- `pxp_run()`: 启动 PXP 作业。使用特定于操作系统的机制来阻止绘图线程。PXP 必须完成绘图才能离开此功能。
- There are configurable area thresholds which are used to decide whether the area will be processed by CPU, or by PXP. Areas smaller than defined value will be processed by CPU, areas bigger than the threshold will be processed by PXP. These thresholds may be defined as a preprocessor variables. Default values are defined in `lv_gpu/lv_gpu_nxp_pxp.h`
 - `GPU_NXP_PXP_BLIT_SIZE_LIMIT`: size threshold for image BLIT, BLIT with color keying, and BLIT with recolor ($OPA > LV_OPA_MAX$)
 - `GPU_NXP_PXP_BLIT_OPA_SIZE_LIMIT`: size threshold for image BLIT and BLIT with color keying with transparency ($OPA < LV_OPA_MAX$)
 - `GPU_NXP_PXP_FILL_SIZE_LIMIT`: size threshold for fill operation ($OPA > LV_OPA_MAX$)
 - `GPU_NXP_PXP_FILL_OPA_SIZE_LIMIT`: size threshold for fill operation with transparency ($OPA < LV_OPA_MAX$)
- 有可配置的区域阈值，用于决定该区域是由 CPU 处理还是由 PXP 处理。小于定义值的区域将由 CPU 处理，大于阈值的区域将由 PXP 处理。这些阈值可以定义为预处理器变量。默认值定义在 `lv_gpu/lv_gpu_nxp_pxp.h`
 - `GPU_NXP_PXP_BLIT_SIZE_LIMIT`: 图像 BLIT、带颜色键控的 BLIT 和带重新着色的 BLIT 的大小阈值 ($OPA > LV_OPA_MAX$)
 - `GPU_NXP_PXP_BLIT_OPA_SIZE_LIMIT`: 图像 BLIT 和 BLIT 的大小阈值，带有透明度的颜色键控 ($OPA < LV_OPA_MAX$)
 - `GPU_NXP_PXP_FILL_SIZE_LIMIT`: 填充操作的大小阈值 ($OPA > LV_OPA_MAX$)
 - `GPU_NXP_PXP_FILL_OPA_SIZE_LIMIT`: 透明填充操作的尺寸阈值 ($OPA < LV_OPA_MAX$)

3.5 Espressif (ESP32)

Since v7.7.1 LVGL includes a Kconfig file, so LVGL can be used as an ESP-IDF v4 component.

由于 v7.7.1 LVGL 包含一个 Kconfig 文件，因此 LVGL 可以用作 ESP-IDF v4 组件。

3.5.1 Get the LVGL demo project for ESP32 (获取 ESP32 的 LVGL 演示项目)

We've created `lv_port_esp32`, a project using ESP-IDF and LVGL to show one of the demos from `lv_examples`. You are able to configure the project to use one of the many supported display controllers, see `lvgl_esp32_drivers` for a complete list of supported display and indev (touch) controllers.

我们创建了 `lv_port_esp32` 项目示例，这是一个使用 ESP-IDF 和 LVGL 的项目，这是从 `lv_examples` 中移植的一个演示。您可以将项目配置为使用众多支持的显示控制器之一，请参阅 `lvgl_esp32_drivers` 以获取支持的显示和 indev (触摸) 控制器的完整列表。

3.5.2 Use LVGL in your ESP32 project (在 ESP32 项目中使用 LVGL)

Prerequisites (前提)

ESP-IDF v4 framework is the suggested version to use.

建议在 ESP-IDF v4 框架上使用 LVGL。

Get LVGL (获取 LVGL)

You are suggested to add LVGL as a "component". This component can be located inside a directory named "components" on your project root directory.

When your project is a git repository you can include LVGL as a git submodule:

建议您将 LVGL 添加为“组件”。该组件可以位于项目根目录下名为“components”的目录中。

当您的项目是 git 存储库时，您可以将 LVGL 作为 git 子模块包含在内：

```
git submodule add https://github.com/lvgl/lvgl.git components/lvgl
```

The above command will clone LVGL's main repository into the `components/lvgl` directory. LVGL includes a `CMakeLists.txt` file that sets some configuration options so you can use LVGL right away.

When you are ready to configure LVGL launch the configuration menu with `idf.py menuconfig` on your project root directory, go to `Component config` and then `LVGL configuration`.

上面的命令会将 LVGL 的主存储库克隆到 `components/lvgl` 目录中。LVGL 包含一个 `CMakeLists.txt` 文件，该文件设置了一些配置选项，因此您可以立即使用 LVGL。

当您准备好配置 LVGL 时，在项目根目录中使用 `idf.py menuconfig` 启动配置菜单，转到 `Component config`，然后转到 `LVGL configuration`。

3.5.3 Use lvgl_esp32_drivers in your project (在您的项目中使用 lvgl_esp32_drivers)

You are suggested to add `lvgl_esp32_drivers` as a "component". This component can be located inside a directory named "components" on your project root directory.

When your project is a git repository you can include `lvgl_esp32_drivers` as a git submodule:

建议您添加 `lvgl_esp32_drivers` 作为“组件”。该组件可以位于项目根目录下名为“components”的目录中。当您的项目是 git 存储库时，您可以将 `lvgl_esp32_drivers` 作为 git 子模块包含在内：

```
git submodule add https://github.com/lvgl/lvgl_esp32_drivers.git components/lvgl_esp32_drivers
```

Support for ESP32-S2 (支持 ESP32-S2)

Basic support for ESP32-S2 has been added into the `lvgl_esp32_drivers` repository.

对 ESP32-S2 的基本支持已添加到 `lvgl_esp32_drivers` 存储库中。

3.6 Arduino

The core [LVGL library](#) and the [demos](#) are directly available as Arduino libraries.

Note that you need to choose a powerful enough board to run LVGL and your GUI. See the [requirements of LVGL](#).

For example ESP32 is a good candidate to create your UI with LVGL.

[LVGL 核心库](#) 和 [演示仓库](#) 可直接作为 Arduino 库使用。

请注意，您需要选择一个足够强大的板来运行 LVGL 和您的 GUI。请参阅 [LVGL 的运行要求](#)。

例如，ESP32 是使用 LVGL 创建 UI 的不错选择。

3.6.1 Get the LVGL Arduino library (获取 LVGL Arduino 库)

LVGL can be installed via the Arduino IDE Library Manager or as a .ZIP library.

LVGL 可以通过 Arduino IDE 库管理器安装或作为.ZIP 库安装。

3.6.2 Set up drivers (设置驱动程序)

To get started it's recommended to use [TFT_eSPI](#) library as a TFT driver to simplify testing. To make it work setup [TFT_eSPI](#) according to your TFT display type via editing either

- `User_Setup.h`
- or by selecting a configuration in the `User_Setup_Select.h`

Both files are located in [TFT_eSPI](#) library's folder.

首先，建议使用 [TFT_eSPI](#) 库作为 TFT 驱动程序以简化测试。为了使其工作，根据您的 TFT 显示类型通过编辑设置 “[TFT_eSPI](#)”

- `User_Setup.h`
- 或通过在 “`User_Setup_Select.h`” 中选择配置这两个文件都位于 [TFT_eSPI](#) 库的文件夹中。

3.6.3 Configure LVGL (配置 LVGL)

LVGL has its own configuration file called `lv_conf.h`. When LVGL is installed the followings needs to be done to configure it:

1. Go to directory of the installed Arduino libraries
2. Go to `lvgl` and copy `lv_conf_template.h` as `lv_conf.h` into the Arduino Libraries directory next to the `lvgl` library folder.
3. Open `lv_conf.h` and change the first `#if 0` to `#if 1`
4. Set the color depth of your display in `LV_COLOR_DEPTH`

5. Set LV_TICK_CUSTOM 1

LVGL 有自己的配置文件，名为 `lv_conf.h`。安装 LVGL 后，需要进行以下配置：

1. 进入已安装的 Arduino 库目录
2. 转到 `lvgl` 并将 `lv_conf_template.h` 作为 `lv_conf.h` 复制到 `lvgl` 库文件夹旁边的 Arduino Libraries 目录中。
3. 打开 `lv_conf.h`，将第一个 `#if 0` 改为 `#if 1`
4. 在 `LV_COLOR_DEPTH` 中设置你显示的颜色深度
5. 设置 `LV_TICK_CUSTOM 1`

3.6.4 Initialize LVGL and run an example (初始化 LVGL 并运行示例)

Take a look at `LVGL_Arduino.ino` to see how to initialize LVGL. TFT_eSPI is used as the display driver.

In the INO file you can see how to register a display and a touch pad for LVGL and call an example.

Note that, there is no dedicated INO file for every example but you can call functions like `lv_example_btn_1()` or `lv_example_slider_1()` to run an example. Most of the examples are available in the `lvgl/examples` folder. Some are also available in `lv_demos`, which needs to be installed and configured separately.

我们看看 `LVGL_Arduino.ino` 是如何初始化 LVGL 的。TFT_eSPI 用作显示驱动程序。

在 INO 文件中，您可以看到如何为 LVGL 注册显示器和触摸板并调用示例。

请注意，每个示例都没有专用的 INO 文件，但您可以调用诸如“`lv_example_btn_1()`”或“`lv_example_slider_1()`”之类的函数来运行示例。大多数示例都可以在 `lvgl/examples` 文件夹中找到。`lv_demos` 中也有一些，需要单独安装配置。

3.6.5 Debugging and logging (调试和日志)

In case of trouble LVGL can display debug information. In the `LVGL_Arduino.ino` example there is `my_print` method, which allow to send this debug information to the serial interface. To enable this feature you have to edit `lv_conf.h` file and enable logging in the section `log settings`:

如果出现故障，LVGL 可以显示调试信息。在 `LVGL_Arduino.ino` 示例中有 `my_print` 方法，它允许将此调试信息发送到串行接口。要启用此功能，您必须编辑“`lv_conf.h`”文件并在“日志设置”部分启用日志记录：

```
/*Log settings*/
#define USE_LV_LOG      1 /*Enable/disable the log module*/
#if LV_USE_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
```

After enabling the log module and setting `LV_LOG_LEVEL` accordingly the output log is sent to the `Serial` port @ 115200 bps.

启用日志模块并相应地设置 `LV_LOG_LEVEL` 后，输出日志会以 115200 bps 的速度发送到“串行”端口。

3.7 Micropython

3.7.1 What is Micropython? (什么是 Micropython?)

Micropython is Python for microcontrollers. Using Micropython, you can write Python3 code and run it even on a bare metal architecture with limited resources.

Micropython 是用于微控制器的 Python。使用 Micropython，即使在资源有限的裸机架构上，您也可以编写 Python3 代码并运行它。

Highlights of Micropython (Micropython 的亮点)

- **Compact** - Fits and runs within just 256k of code space and 16k of RAM. No OS is needed, although you can also run it with an OS, if you want.
- **Compatible** - Strives to be as compatible as possible with normal Python (known as CPython).
- **Versatile** - Supports many architectures (x86, x86-64, ARM, ARM Thumb, Xtensa).
- **Interactive** - No need for the compile-flash-boot cycle. With the REPL (interactive prompt) you can type commands and execute them immediately, run scripts etc.
- **Popular** - Many platforms are supported. The user base is growing bigger. Notable forks: [MicroPython](#), [CircuitPython](#), [MicroPython_ESP32_psRAM_LoBo](#)
- **Embedded Oriented** - Comes with modules specifically for embedded systems, such as the `machine module` for accessing low-level hardware (I/O pins, ADC, UART, SPI, I2C, RTC, Timers etc.)

-
- **紧凑** - 适合并在仅 256k 的代码空间和 16k 的 RAM 内运行。不需要操作系统，但如果需要，您也可以使用操作系统运行它。
 - **兼容** - 努力与普通 Python（称为 CPython）尽可能兼容。
 - **多功能** - 支持多种架构（x86、x86-64、ARM、ARM Thumb、Xtensa）。
 - **交互式** - 不需要编译-闪存-启动循环。使用 REPL（交互式提示），您可以键入命令并立即执行它们、运行脚本等。
 - **流行** - 支持许多平台。用户群越来越大。值得注意的分支：[MicroPython](#)、[CircuitPython](#)、[MicroPython_ESP32_psRAM_LoBo](#) /[MicroPython_ESP32_psRAM_LoBo](#)
 - **Embedded Oriented** - 带有专门用于嵌入式系统的模块，例如 `machine module`，用于访问低级硬件（I/O 引脚、ADC、UART、SPI、I2C、RTC、定时器等）
-

3.7.2 Why Micropython + LVGL? (为什么是 Micropython + LVGL?)

Currently, Micropython does not have a good high-level GUI library by default. LVGL is an Object Oriented Component Based high-level GUI library, which seems to be a natural candidate to map into a higher level language, such as Python. LVGL is implemented in C and its APIs are in C.

目前，Micropython 没有一个好的高级 GUI 库 默认情况下。LVGL 是一个 Object Oriented Component Based 高级 GUI 库，它似乎是映射到更高级别的自然候选者级别语言，如 Python。LVGL 是用 C 实现的，它的 API 是用 C 编写的。

Here are some advantages of using LVGL in Micropython: (以下是在 Micropython 中使用 LVGL 的一些优势:)

- Develop GUI in Python, a very popular high level language. Use paradigms such as Object Oriented Programming.
- Usually, GUI development requires multiple iterations to get things right. With C, each iteration consists of **Change code > Build > Flash > Run**. In Micropython it's just **Change code > Run**! You can even run commands interactively using the **REPL** (the interactive prompt)
- 用 Python 开发 GUI，这是一种非常流行的高级语言。使用诸如面向对象编程之类的范式。
- 通常，GUI 开发需要多次迭代才能把事情做好。使用 C，每次迭代都包含 **Change code > Build > Flash > Run**。在 Micropython 中，它只是 **Change code > Run**！您甚至可以使用 **REPL**（交互式提示）以交互方式运行命令

Micropython + LVGL could be used for: (Micropython + LVGL 可用于:)

- Fast prototyping GUI.
- Shortening the cycle of changing and fine-tuning the GUI.
- Modelling the GUI in a more abstract way by defining reusable composite objects, taking advantage of Python's language features such as Inheritance, Closures, List Comprehension, Generators, Exception Handling, Arbitrary Precision Integers and others.
- Make LVGL accessible to a larger audience. No need to know C in order to create a nice GUI on an embedded system. This goes well with [CircuitPython vision](#). CircuitPython was designed with education in mind, to make it easier for new or unexperienced users to get started with embedded development.
- Creating tools to work with LVGL at a higher level (e.g. drag-and-drop designer).

-
- 快速原型设计 GUI。
 - 缩短更改和微调 GUI 的周期。
 - 通过定义可重用的复合对象，利用 Python 的语言特性，如继承、闭包、列表理解、生成器、异常处理、任意精度整数等，以更抽象的方式对 GUI 建模。
 - 让更多的观众可以访问 LVGL。无需了解 C 即可在嵌入式系统上创建漂亮的 GUI。这与 [CircuitPython 愿景](#) 相得益彰。CircuitPython 的设计考虑了教育，使新用户或没有经验的用户更容易开始嵌入式开发。
 - 创建工具以在更高级别使用 LVGL（例如拖放设计器）。
-

3.7.3 So what does it look like? (它看起来像什么?)

TL;DR: It's very much like the C API, but Object Oriented for LVGL components.

Let's dive right into an example!

TL;DR: 它非常像 C API，但是 LVGL 组件是面向对象的。

让我们直接进入一个例子！

A simple example (一个简单的例子)

```
import lvgl as lv
lv.init()
scr = lv.obj()
btn = lv.btn(scr)
btn.align(lv.scr_act(), lv.ALIGN.CENTER, 0, 0)
label = lv.label(btn)
label.set_text("Button")
lv.scr_load(scr)
```

3.7.4 How can I use it? (我怎样才能使用它?)

Online Simulator (在线模拟器)

If you want to experiment with LVGL + Micropython without downloading anything - you can use our online simulator! It's a fully functional LVGL + Micropython that runs entirely in the browser and allows you to edit a python script and run it.

[Click here to experiment on the online simulator](#)

Hello World

Note: the online simulator is available for lvgl v6 and v7.

如果你想在不下载任何东西的情况下试验 LVGL + Micropython - 你可以使用我们的在线模拟器！它是一个功能齐全的 LVGL + Micropython，完全在浏览器中运行，并允许您编辑 Python 脚本并运行它。

[【点击这里体验在线模拟器上实验】 \(<https://sim.lvgl.io/>\)](#)

Hello World

注意：此在线模拟器适用于 lvgl v6 和 v7。

PC Simulator (PC 模拟器)

Micropython is ported to many platforms. One notable port is "unix", which allows you to build and run Micropython (+LVGL) on a Linux machine. (On a Windows machine you might need Virtual Box or WSL or MinGW or Cygwin etc.)

[Click here to know more information about building and running the unix port](#)

Micropython 被移植到许多平台。一个值得注意的端口是“unix”，它允许您在 Linux 机器上构建和运行 Micropython (+LVGL)。(在 Windows 机器上，您可能需要 Virtual Box 或 WSL 或 MinGW 或 Cygwin 等)

[点击这里了解更多关于构建和运行 unix 端口的信息](#)

Embedded platform (嵌入式平台)

In the end, the goal is to run it all on an embedded platform. Both Micropython and LVGL can be used on many embedded architectures, such as stm32, ESP32 etc. You would also need display and input drivers. We have some sample drivers (ESP32+ILI9341, as well as some other examples), but chances are you would want to create your own input/display drivers for your specific hardware. Drivers can be implemented either in C as a Micropython module, or in pure Micropython!

最后，目标是在嵌入式平台上运行它。Micropython 和 LVGL 都可以用于许多嵌入式架构，例如 stm32、ESP32 等。您还需要显示和输入驱动程序。我们有一些示例驱动程序 (ESP32+ILI9341，以及其他一些示例)，但您可能希望为您的特定硬件创建自己的输入/显示驱动程序。驱动程序可以在 C 中实现为 Micropython 模块，也可以在纯 Micropython 中实现！

3.7.5 Where can I find more information? (我在哪里可以找到更多的信息?)

- In this Blog Post
- [lv_micropython README](#)
- [lv_binding_micropython README](#)
- The [LVGL micropython forum](#) (Feel free to ask anything!)
- At Micropython: [docs](#) and [forum](#)
- 在这个博客帖子
- [lv_micropython 自述文件](#)
- [lv_binding_micropython 自述文件](#)
- [LVGL micropython 论坛](#) (请随意提问!)
- 在 Micropython: [docs](#) 和 [forum](#)

3.8 NuttX RTOS

3.8.1 What is NuttX? (什么是 NuttX?)

NuttX is a mature and secure real-time operating system (RTOS) with an emphasis on technical standards compliance and small size. It is scalable from 8-bit to 64-bit microcontrollers and microprocessors and compliant with the Portable Operating System Interface (POSIX) and the American National Standards Institute (ANSI) standards and with many Linux-like subsystems. The best way to think about NuttX is to think of it as a small Unix/Linux for microcontrollers.

NuttX 是一个成熟且安全的实时操作系统 (RTOS)，强调技术标准合规性和小尺寸。它可以从 8 位扩展到 64 位微控制器和微处理器，并符合便携式操作系统接口 (POSIX) 和美国国家标准协会 (ANSI) 标准以及许多类似 Linux 的子系统。考虑 NuttX 的最佳方式是将其视为用于微控制器的小型 Unix/Linux。

Highlights of NuttX (NuttX 的亮点)

- **Small** - Fits and runs in microcontrollers as small as 32KB Flash and 8KB of RAM.
- **Compliant** - Strives to be as compatible as possible with POSIX and Linux.
- **Versatile** - Supports many architectures (ARM, ARM Thumb, AVR, MIPS, OpenRISC, RISC-V 32-bit and 64-bit, RX65N, x86-64, Xtensa, Z80/Z180, etc.).
- **Modular** - Its modular design allows developers to select only what really matters and use modules to include new features.
- **Popular** - NuttX is used by many companies around the world. Probably you already used a product with NuttX without knowing it was running NuttX.
- **Predictable** - NuttX is a preemptible Realtime kernel, so you can use it to create predictable applications for realtime control.

-
- 小 - 适合并在小至 32KB 闪存和 8KB RAM 的微控制器中运行。
 - Compliant - 力求尽可能与 POSIX 和 Linux 兼容。
 - 多功能 - 支持多种架构 (ARM、ARM Thumb、AVR、MIPS、OpenRISC、RISC-V 32 位和 64 位、RX65N、x86-64、Xtensa、Z80/Z180 等)。

- **模块化** - 其模块化设计允许开发人员仅选择真正重要的内容并使用模块来包含新功能。
 - **流行** - NuttX 被世界各地的许多公司使用。可能您已经使用 NuttX 的产品，但不知道它正在运行 NuttX。
 - **Predictable** - NuttX 是一个抢占式实时内核，因此您可以使用它来创建可预测的应用程序以进行实时控制。
-

3.8.2 Why NuttX + LVGL? (为什么是 NuttX + LVGL ?)

Although NuttX has its own graphic library called **NX**, LVGL is a good alternative because users could find more eye-candy demos and they can reuse code from previous projects. LVGL is an **Object Oriented Component Based** high-level GUI library, that could fit very well for a RTOS with advanced features like NuttX. LVGL is implemented in C and its APIs are in C.

尽管 NuttX 有自己的图形库，称为 **NX**，但 LVGL 是一个不错的选择，因为用户可以找到更多吸引眼球的演示和他们可以重用以前项目中的代码。LVGL 是一个 **Object Oriented Component Based** 高级 GUI 库，它非常适合具有高级功能的 RTOS，例如坚果 X。LVGL 是用 C 实现的，它的 API 是用 C 编写的。

Here are some advantages of using LVGL in NuttX (以下是在 NuttX 中使用 LVGL 的一些优势)

- Develop GUI in Linux first and when it is done just compile it for NuttX. Nothing more, no wasting of time.
- Usually, GUI development for low level RTOS requires multiple iterations to get things right, where each iteration consists of **Change code > Build > Flash > Run**. Using LVGL, Linux and NuttX you can reduce this process and just test everything on your computer and when it is done, compile it on NuttX and that is it.
- 首先在 Linux 中开发 GUI，完成后只需为 NuttX 编译它。不多说了，不浪费时间了。
- 通常，低级 RTOS 的 GUI 开发需要多次迭代才能使事情正确，其中每次迭代包括 ** 更改代码 > 构建 > Flash > 运行 **。使用 LVGL、Linux 和 NuttX，您可以减少这个过程，只需在您的计算机上测试所有内容，完成后，在 NuttX 上编译它，就是这样。

NuttX + LVGL could be used for (NuttX + LVGL 可用于)

- GUI demos to demonstrate your board graphics capacities.
 - Fast prototyping GUI for MVP (Minimum Viable Product) presentation.
 - visualize sensor data directly and easily on the board without using a computer.
 - Final products with a GUI without a touchscreen (i.e. 3D Printer Interface using Rotary Encoder to Input data).
 - Final products with a touchscreen (and all sorts of bells and whistles).
-

- GUI 演示来展示您的电路板图形能力。
 - MVP（最小可行产品）演示的快速原型设计 GUI。
 - 无需使用计算机即可在板上直接轻松地可视化传感器数据。
 - 带 GUI 且不带触摸屏的最终产品（即使用旋转编码器输入数据的 3D 打印机界面）。
 - 带有触摸屏的最终产品（以及各种花里胡哨的东西）。
-

3.8.3 How to get started with NuttX and LVGL? (如何开始使用 NuttX 和 LVGL?)

There are many boards in the NuttX mainline (<https://github.com/apache/incubator-nuttx>) with support for LVGL. Let's use the **STM32F429IDISCOVERY** as example because it is a very popular board.

NuttX 主线 (<https://github.com/apache/incubator-nuttx>) 中有很多支持 LVGL 的开发板。我们以 **STM32F429IDISCOVERY** 为例，因为它是一款非常受欢迎的开发板。

First you need to install the pre-requisite on your system (首先，您需要在系统上安装依赖)

Let's use the Windows Subsystem for Linux

让我们使用适用于 Linux 的 Windows 子系统

```
$ 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
```

Now let's to create a workspace to save our files (现在让我们创建一个工作区来保存我们的文件)

```
$ mkdir ~/nuttxspace
$ cd ~/nuttxspace
```

Clone the NuttX and Apps repositories: (克隆 NuttX 和 Apps 存储库:)

```
$ git clone https://github.com/apache/incubator-nuttx nuttx
$ git clone https://github.com/apache/incubator-nuttx-apps apps
```

Configure NuttX to use the stm32f429i-disco board and the LVGL Demo (配置 NuttX 以使用 stm32f429i-disco 板和 LVGL Demo)

```
$ ./tools/configure.sh stm32f429i-disco:lvgl
$ make
```

If everything went fine you should have now the file **nuttx.bin** to flash on your board:

如果一切顺利，你现在应该有文件 **nuttx.bin** 在你的板上的 LED 灯闪烁：

```
$ ls -l nuttx.bin
-rwxrwxr-x 1 alan alan 287144 Jun 27 09:26 nuttx.bin
```

Flashing the firmware in the board using OpenOCD: (使用 OpenOCD 刷新板上的固件:)

```
$ sudo openocd -f interface/stlink-v2.cfg -f target/stm32f4x.cfg -c init -c "reset  
halt" -c "flash write_image erase nuttx.bin 0x08000000"
```

Reset the board and using the 'NSH>' terminal start the LVGL demo:

重启开发板并使用 “NSH>” 终端启动 LVGL 演示:

```
nsh> lvglDemo
```

3.8.4 Where can I find more information? (我在哪里可以找到更多信息 ?)

- This blog post: [LVGL on LPCXpresso54628](#)
- NuttX mailing list: [Apache NuttX Mailing List](#)
- 这篇博文: [LPCXpresso54628 上的 LVGL](#)
- NuttX 邮件列表: [Apache NuttX 邮件列表](#)

CHAPTER 4

Porting (移植)

4.1 Set-up a project (设置项目)

4.1.1 Get the library (获取 LVGL 图形库)

LVGL is available on GitHub: <https://github.com/lvgl/lvgl>.

You can clone it or download the latest version of the library from GitHub.

The graphics library itself is the **lvgl** directory which should be copied into your project.

LVGL 可在 GitHub 上获得: <https://github.com/lvgl/lvgl>。

您可以克隆它或从 GitHub 下载最新版本的库。

图形库本身是 **lvgl** 目录，应将其复制到您的项目中。

4.1.2 Configuration file (修改配置文件)

There is a configuration header file for LVGL called **lv_conf.h**. In this you can set the library's basic behavior, disable unused modules and features, adjust the size of memory buffers in compile-time, etc.

Copy **lvgl/lv_conf_template.h** next to the *lvgl* directory and rename it to *lv_conf.h*. Open the file and change the `#if 0` at the beginning to `#if 1` to enable its content.

lv_conf.h can be copied to another place as well but then you should add **LV_CONF_INCLUDE_SIMPLE** define to your compiler options (e.g. `-DLV_CONF_INCLUDE_SIMPLE` for gcc compiler) and set the include path manually. In this case LVGL will attempt to include `lv_conf.h` simply with `#include "lv_conf.h"`.

In the config file comments explain the meaning of the options. Be sure to set at least **LV_COLOR_DEPTH** according to your display's color depth.

有一个名为 **lv_conf.h** 的 LVGL 配置头文件。在这里，您可以设置库的基本行为、禁用未使用的模块和功能、在编译时调整内存缓冲区的大小等。

复制 `lvgl` 目录旁边的 `lvgl/lv_conf_template.h` 并将其重命名为 `lv_conf.h`。打开文件并将开头的 “#if 0” 更改为 “#if 1” 以启用其内容。

`lv_conf.h` 也可以复制到另一个地方，但是你应该添加 `LV_CONF_INCLUDE_SIMPLE` 定义到你的编译器选项（例如 `-DLV_CONF_INCLUDE_SIMPLE` 用于 gcc 编译器）并手动设置包含路径。在这种情况下，LVGL 将尝试使用 `#include "lv_conf.h"` 简单地包含 `lv_conf.h`。

在配置文件的注释中解释了选项的含义。请务必根据显示器的颜色深度至少设置 “`LV_COLOR_DEPTH`”。

4.1.3 Initialization (初始化)

To use the graphics library you have to initialize it and the other components too. The order of the initialization is:

1. Call `lv_init()`.
2. Initialize your drivers.
3. Register the display and input devices drivers in LVGL. Learn more about [Display](#) and [Input device](#) registration.
4. Call `lv_tick_inc(x)` every X milliseconds in an interrupt to tell the elapsed time. [Learn more](#).
5. Call `lv_timer_handler()` every few milliseconds to handle LVGL related tasks. [Learn more](#).

要使用图形库，您必须初始化它和其他组件。初始化的顺序是：

1. 调用 `lv_init()`。
2. 初始化您的驱动程序。
3. 在 LVGL 中注册显示和输入设备驱动程序。详细了解 [Display](#) 和 [Input device](#) 注册。
4. 在中断中每隔 x 毫秒调用 `lv_tick_inc(x)` 以告知经过的时间。了解更多。
5. 每隔几毫秒调用 `lv_timer_handler()` 来处理 LVGL 相关的任务。了解更多。

4.2 Display interface (显示接口)

To register a display for LVGL a `lv_disp_draw_buf_t` and a `lv_disp_drv_t` variable have to be initialized.

- `lv_disp_draw_buf_t` contains internal graphic buffer(s) called draw buffer(s).
- `lv_disp_drv_t` contains callback functions to interact with the display and manipulate drawing related things.

要为 LVGL 注册一个显示器，必须初始化一个 `lv_disp_draw_buf_t` 和一个 `lv_disp_drv_t` 变量。

- `lv_disp_draw_buf_t` 包含称为绘制缓冲区的内部图形缓冲区。
- `lv_disp_drv_t` 包含与显示交互和操作绘图相关事物的回调函数。

4.2.1 Draw buffer (绘制缓冲区)

Draw buffer(s) are simple array(s) that LVGL uses to render the content of the screen. Once rendering is ready the content of the draw buffer is sent to the display using the `flush_cb` function set in the display driver (see below).

A draw draw buffer can be initialized via a `lv_disp_draw_buf_t` variable like this:

绘制缓冲区是 LVGL 用来渲染屏幕内容的简单数组。一旦渲染准备就绪，绘制缓冲区的内容将使用显示驱动程序中设置的 `flush_cb` 函数发送到显示器（见下文）。

绘制绘制缓冲区可以通过 “`lv_disp_draw_buf_t`” 变量初始化，如下所示：

```

/*A static or global variable to store the buffers*/
static lv_disp_draw_buf_t disp_buf;

/*Static or global buffer(s). The second buffer is optional*/
static lv_color_t buf_1[MY_DISP_HOR_RES * 10];
static lv_color_t buf_2[MY_DISP_HOR_RES * 10];

/*Initialize `disp_buf` with the buffer(s). With only one buffer use NULL instead buf_
→2 */
lv_disp_draw_buf_init(&disp_buf, buf_1, buf_2, MY_DISP_HOR_RES*10);

```

Note that `lv_disp_draw_buf_t` needs to be static, global or dynamically allocated and not a local variable destroyed if goes out of the scope.

As you can see the draw buffer can be smaller than the screen. In this case, the larger areas will be redrawn in smaller parts that fit into the draw buffer(s). If only a small area changes (e.g. a button is pressed) then only that area will be refreshed.

A larger buffer results in better performance but above 1/10 screen sized buffer(s) there is no significant performance improvement. Therefore it's recommended to choose the size of the draw buffer(s) to at least 1/10 screen sized.

If only **one buffer** is used LVGL draws the content of the screen into that draw buffer and sends it to the display. This way LVGL needs to wait until the content of the buffer is sent to the display before drawing something new in it.

If **two buffers** are used LVGL can draw into one buffer while the content of the other buffer is sent to display in the background. DMA or other hardware should be used to transfer the data to the display to let the MCU draw meanwhile. This way, the rendering and refreshing of the display become parallel.

请注意, `lv_disp_draw_buf_t` 需要是静态的、全局的或动态分配的, 而不是超出范围时销毁的局部变量。如您所见, 绘制缓冲区可以小于屏幕。在这种情况下, 较大的区域将被重新绘制为适合绘制缓冲区的较小部分。如果只有一个小区域发生变化 (例如按下按钮), 则只会刷新该区域。

更大的缓冲区会导致更好的性能, 但超过 1/10 屏幕大小的缓冲区没有显着的性能改进。因此, 建议选择绘制缓冲区的大小至少为屏幕大小的 1/10。

如果只使用一个缓冲区, LVGL 将屏幕内容绘制到该绘制缓冲区中并将其发送到显示器。这样 LVGL 需要等到缓冲区的内容发送到显示器, 然后再在其中绘制新内容。

如果使用两个缓冲区, LVGL 可以绘制到一个缓冲区中, 而另一个缓冲区的内容被发送到后台显示。应使用 DMA 或其他硬件将数据传输到显示器, 让 MCU 同时绘制。这样, 显示的渲染和刷新变得并行。

In the display driver (`lv_disp_drv_t`) the `full_refresh` bit can be enabled to force LVGL to always redraw the whole screen. This works in both *one buffer* and *two buffers* modes.

If `full_refresh` is enabled and 2 screen sized draw buffers are provided, LVGL's display handling works like "traditional" double buffering. This means in `flush_cb` only the address of the frame buffer needs to be changed to the provided pointer (`color_p` parameter). This configuration should be used if the MCU has LCD controller periphery and not with an external display controller (e.g. ILI9341 or SSD1963).

You can measure the performance of different draw buffer configurations using the [benchmark example](#).

在显示驱动程序 (`lv_disp_drv_t`) 中, 可以启用 `full_refresh` 位以强制 LVGL 始终重绘整个屏幕。这适用于 *one buffer* 和 *two buffers* 模式。

如果启用 `full_refresh` 并提供 2 个屏幕大小的绘制缓冲区, LVGL 的显示处理就像“传统”双缓冲一样工作。

这意味着在 `flush_cb` 中只有帧缓冲区的地址需要更改为提供的指针 (`color_p` 参数)。如果 MCU 具有 LCD 控制器外围设备而不是外部显示控制器 (例如 ILI9341 或 SSD1963), 则应使用此配置。

您可以使用 [基准示例](#) 测量不同绘制缓冲区配置的性能。

4.2.2 Display driver (显示驱动程序)

Once the buffer initialization is ready a `lv_disp_drv_t` display driver needs to be

1. initialized with `lv_disp_drv_init(&disp_drv)`
2. its fields need to be set
3. it needs to be registered in LVGL with `lv_disp_drv_register(&disp_drv)`

Note that `lv_disp_drv_t` also needs to be static, global or dynamically allocated and not a local variable destroyed if goes out of the scope.

一旦缓冲区初始化准备好，`lv_disp_drv_t` 显示驱动程序需要

1. 用 `lv_disp_drv_init(&disp_drv)` 初始化
2. 它的字段需要设置
3. 需要在 LVGL 中用 `lv_disp_drv_register(&disp_drv)` 注册

请注意，`lv_disp_drv_t` 也需要是静态的、全局的或动态分配的，而不是超出范围时销毁的局部变量。

Mandatory fields (必须要适配的部分)

In the most simple case only the following fields of `lv_disp_drv_t` need to be set:

- `draw_buf` pointer to an initialized `lv_disp_draw_buf_t` variable.
- `hor_res` horizontal resolution of the display in pixels.
- `ver_res` vertical resolution of the display in pixels.
- `flush_cb` a callback function to copy a buffer's content to a specific area of the display. `lv_disp_flush_ready(&disp_drv)` needs to be called when flushing is ready. LVGL might render the screen in multiple chunks and therefore call `flush_cb` multiple times. To see if the current one is the last chunk of rendering use `lv_disp_flush_is_last(&disp_drv)`.

在最简单的情况下，只需要设置 `lv_disp_drv_t` 的以下字段：

- 指向初始化的 `lv_disp_draw_buf_t` 变量的 `draw_buf` 指针。
- `hor_res` 显示器的水平分辨率（以像素为单位）。
- `ver_res` 显示器的垂直分辨率（以像素为单位）。
- `flush_cb` 一个回调函数，用于将缓冲区的内容复制到显示器的特定区域。

`lv_disp_flush_ready(&disp_drv)` 需要在刷新准备好时调用。LVGL 可能会以多个块呈现屏幕，因此多次调用 `flush_cb`。要查看当前是否是渲染的最后一个块，请使用 `lv_disp_flush_is_last(&disp_drv)`。

Optional fields (可选的部分)

There are some optional data fields:

- `color_chroma_key` A color which will be drawn as transparent on chrome keyed images. Set to `LV_COLOR_CHROMA_KEY` by default from `lv_conf.h`.
- `anti_aliasing` use anti-aliasing (edge smoothing). Enabled by default if `LV_COLOR_DEPTH` is set to at least 16 in `lv_conf.h`.
- `rotated` and `sw_rotate` See the [Rotation](#) section below.
- `screen_transp` if 1 the screen itself can have transparency as well. `LV_COLOR_SCREEN_TRANSP` needs to be enabled in `lv_conf.h` and requires `LV_COLOR_DEPTH` 32.
- `user_data` A custom `void` user data for the driver..

有一些可选的数据字段:

- `color_chroma_key` 将在镀铬键控图像上绘制为透明的颜色。从 `lv_conf.h` 默认设置为 `LV_COLOR_CHROMA_KEY`。
- `anti_aliasing` 使用抗锯齿（边缘平滑）。如果在 `lv_conf.h` 中将 `LV_COLOR_DEPTH` 设置为至少 16，则默认启用。
- `rotated` 和 `sw_rotate` 请参阅下面的[Rotation](#) 部分。
- `screen_transp` 如果 1 屏幕本身也可以具有透明度。`LV_COLOR_SCREEN_TRANSP` 需要在 `lv_conf.h` 中启用并且需要 `LV_COLOR_DEPTH` 32。
- `user_data` 驱动程序的自定义 `void` 用户数据..

Some other optional callbacks to make easier and more optimal to work with monochrome, grayscale or other non-standard RGB displays:

- `rounder_cb` Round the coordinates of areas to redraw. E.g. a 2x2 px can be converted to 2x8. It can be used if the display controller can refresh only areas with specific height or width (usually 8 px height with monochrome displays).
- `set_px_cb` a custom function to write the draw buffer. It can be used to store the pixels more compactly in the draw buffer if the display has a special color format. (e.g. 1-bit monochrome, 2-bit grayscale etc.) This way the buffers used in `lv_disp_draw_buf_t` can be smaller to hold only the required number of bits for the given area size. Note that, rendering with `set_px_cb` is slower than normal rendering.
- `monitor_cb` A callback function that tells how many pixels were refreshed in how much time. Called when the last chunk is rendered and sent to the display.
- `clean_dcache_cb` A callback for cleaning any caches related to the display.

一些其他可选的回调，使处理单色、灰度或其他非标准 RGB 显示器更容易、更优化：

- `rounder_cb` 四舍五入要重绘的区域的坐标。例如。2x2 px 可以转换为 2x8。如果显示控制器只能刷新具有特定高度或宽度的区域（单色显示器通常为 8 像素高度），则可以使用它。
- `set_px_cb` 一个自定义函数来写入绘制缓冲区。如果显示器具有特殊的颜色格式，它可用于将像素更紧凑地存储在绘图缓冲区中。（例如 1 位单色、2 位灰度等）这样，`lv_disp_draw_buf_t` 中使用的缓冲区可以更小，以仅容纳给定区域大小所需的位数。请注意，使用 `set_px_cb` 渲染比普通渲染慢。
- `monitor_cb` 一个回调函数，告诉我们在多长时间内刷新了多少像素。当最后一个块被渲染并发送到显示器时调用。
- `clean_dcache_cb` 用于清理与显示相关的任何缓存的回调。

LVGL has built-in support to several GPUs (see `lv_conf.h`) but if something else is required these functions can be used to make LVGL use a GPU:

- `gpu_fill_cb` fill an area in the memory with a color.
- `gpu_wait_cb` if any GPU function returns while the GPU is still working, LVGL will use this function when required to make sure GPU rendering is ready.

LVGL 内置了对多个 GPU 的支持（参见 `lv_conf.h`），但如果需要其他功能，这些函数可用于使 LVGL 使用 GPU：

- `gpu_fill_cb` 用颜色填充内存中的一个区域。
- `gpu_wait_cb` 如果在 GPU 仍在工作时任何 GPU 函数返回，LVGL 将在需要时使用此函数以确保 GPU 渲染准备就绪。

Examples (示例)

All together it looks like this:

放在一起看起来像这样：

```
static lv_disp_drv_t disp_drv;           /*A variable to hold the drivers. Must beu
↳ static or global.*/
lv_disp_drv_init(&disp_drv);           /*Basic initialization*/
disp_drv.draw_buf = &disp_buf;          /*Set an initialized buffer*/
disp_drv.flush_cb = my_flush_cb;         /*Set a flush callback to draw to theu
↳ display*/
disp_drv.hor_res = 320;                 /*Set the horizontal resolution in pixels*/
disp_drv.ver_res = 240;                 /*Set the vertical resolution in pixels*/
```

```
lv_disp_t * disp;
disp = lv_disp_drv_register(&disp_drv); /*Register the driver and save the createdu
↳ display objects*/
```

Here are some simple examples of the callbacks:

以下是回调的一些简单示例：

```
void my_flush_cb(lv_disp_drv_t * disp_drv, const lv_area_t * area, lv_color_t * color_
↳ p)
{
    /*The most simple case (but also the slowest) to put all pixels to the screen one-
↳ by-one
     *`put_px` is just an example, it needs to implemented by you.*/
    int32_t x, y;
    for(y = area->y1; y <= area->y2; y++) {
        for(x = area->x1; x <= area->x2; x++) {
            put_px(x, y, *color_p)
            color_p++;
        }
    }

    /* IMPORTANT!!!
     * Inform the graphics library that you are ready with the flushing*/
    lv_disp_flush_ready(disp_drv);
}

void my_gpu_fill_cb(lv_disp_drv_t * disp_drv, lv_color_t * dest_buf, const lv_area_t
↳ * dest_area, const lv_area_t * fill_area, lv_color_t color);
{
    /*It's an example code which should be done by your GPU*/
```

(下页继续)

(续上页)

```

uint32_t x, y;
dest_buf += dest_width * fill_area->y1; /*Go to the first line*/

for(y = fill_area->y1; y < fill_area->y2; y++) {
    for(x = fill_area->x1; x < fill_area->x2; x++) {
        dest_buf[x] = color;
    }
    dest_buf+=dest_width; /*Go to the next line*/
}

void my_rounder_cb(lv_disp_drv_t * disp_drv, lv_area_t * area)
{
    /* Update the areas as needed.
     * For example it makes the area to start only on 8th rows and have Nx8 pixel
     * height.*/
    area->y1 = area->y1 & 0x07;
    area->y2 = (area->y2 & 0x07) + 8;
}

void my_set_px_cb(lv_disp_drv_t * disp_drv, uint8_t * buf, lv_coord_t buf_w, lv_coord_
    t x, lv_coord_t y, lv_color_t color, lv_opa_t opa)
{
    /* Write to the buffer as required for the display.
     * For example it writes only 1-bit for monochrome displays mapped vertically.*/
    buf += buf_w * (y >> 3) + x;
    if(lv_color_brightness(color) > 128) (*buf) |= (1 << (y % 8));
    else (*buf) &= ~(1 << (y % 8));
}

void my_monitor_cb(lv_disp_drv_t * disp_drv, uint32_t time, uint32_t px)
{
    printf("%d px refreshed in %d ms\n", time, ms);
}

void my_clean_dcache_cb(lv_disp_drv_t * disp_drv, uint32)
{
    /* Example for Cortex-M (CMSIS) */
    SCB_CleanInvalidateDCache();
}

```

4.2.3 Rotation (旋转屏幕)

LVGL supports rotation of the display in 90 degree increments. You can select whether you'd like software rotation or hardware rotation.

If you select software rotation (`sw_rotate` flag set to 1), LVGL will perform the rotation for you. Your driver can and should assume that the screen width and height have not changed. Simply flush pixels to the display as normal. Software rotation requires no additional logic in your `flush_cb` callback.

There is a noticeable amount of overhead to performing rotation in software, which is why hardware rotation is also available. In this mode, LVGL draws into the buffer as though your screen now has the width and height inverted. You are responsible for rotating the provided pixels yourself.

LVGL 支持以 90 度为增量旋转显示器。您可以选择是要软件轮换还是硬件轮换。

如果您选择软件旋转 (`sw_rotate` 标志设置为 1)，LVGL 将为您执行旋转。您的驱动程序可以并且应该假设屏幕宽度和高度没有改变。只需像往常一样将像素刷新到显示器即可。软件轮换在您的 `flush_cb` 回调中不需要额外的逻辑。

在软件中执行轮换需要大量的开销，这就是硬件轮换也可用的原因。在这种模式下，LVGL 将绘制到缓冲区中，就好像您的屏幕现在具有反转的宽度和高度一样。您有责任自己旋转提供的像素。

The default rotation of your display when it is initialized can be set using the `rotated` flag. The available options are `LV_DISP_ROT_NONE`, `LV_DISP_ROT_90`, `LV_DISP_ROT_180`, or `LV_DISP_ROT_270`. The rotation values are relative to how you would rotate the physical display in the clockwise direction. Thus, `LV_DISP_ROT_90` means you rotate the hardware 90 degrees clockwise, and the display rotates 90 degrees counterclockwise to compensate.

(Note for users upgrading from 7.10.0 and older: these new rotation enum values match up with the old 0/1 system for rotating 90 degrees, so legacy code should continue to work as expected. Software rotation is also disabled by default for compatibility.)

Display rotation can also be changed at runtime using the `lv_disp_set_rotation(disp, rot)` API.

Support for software rotation is a new feature, so there may be some glitches/bugs depending on your configuration. If you encounter a problem please open an issue on [GitHub](#).

初始化时显示器的默认旋转可以使用 `rotated` 标志设置。可用的选项是 “`LV_DISP_ROT_NONE`”、“`LV_DISP_ROT_90`”、“`LV_DISP_ROT_180`” 或 “`LV_DISP_ROT_270`”。旋转值与顺时针方向旋转物理显示器的方式有关。因此，`LV_DISP_ROT_90` 表示您将硬件顺时针旋转 90 度，显示器逆时针旋转 90 度以进行补偿。

(请注意从 7.10.0 及更早版本升级的用户：这些新的旋转枚举值与旧的 0/1 系统匹配，用于旋转 90 度，因此遗留代码应继续按预期工作。默认情况下，软件旋转也被禁用以实现兼容性。)

也可以在运行时使用 `lv_disp_set_rotation(disp, rot)` API 更改显示旋转。

支持软件轮换是一项新功能，因此根据您的配置可能存在一些故障/错误。如果您遇到问题，请在 [GitHub](#) 上打开一个问题。

4.2.4 Further reading (深入学习)

- `lv_port_disp_template.c` for a template for your own driver.
- [Drawing](#) to learn more about how rendering works in LVGL.
- [Display features](#) to learn more about higher level display features.
- `lv_port_disp_template.c` 用于您自己的驱动程序的模板。
- [Drawing](#) 了解更多关于渲染在 LVGL 中是如何工作的。
- [显示功能](#) 以了解有关更高级别显示功能的更多信息。

4.2.5 API

@description Display Driver HAL interface header file

Typedefs

typedef struct *lv_disp_drv_t* **lv_disp_drv_t**

Display Driver structure to be registered by HAL. Only its pointer will be saved in **lv_disp_t** so it should be declared as `static lv_disp_drv_t my_drv` or allocated dynamically.

typedef struct *lv_disp_t* **lv_disp_t**

Display structure.

注解: `lv_disp_drv_t` should be the first member of the structure.

Enums

enum **lv_disp_rot_t**

Values:

- enumerator **LV_DISP_ROT_NONE**
- enumerator **LV_DISP_ROT_90**
- enumerator **LV_DISP_ROT_180**
- enumerator **LV_DISP_ROT_270**

Functions

void **lv_disp_drv_init**(*lv_disp_drv_t* *driver)

Initialize a display driver with default values. It is used to have known values in the fields and not junk in memory. After it you can safely set only the fields you need.

参数 **driver** -- pointer to driver variable to initialize

void **lv_disp_draw_buf_init**(*lv_disp_draw_buf_t* *draw_buf, void *buf1, void *buf2, uint32_t size_in_px_cnt)

Initialize a display buffer

参数

- **draw_buf** -- pointer *lv_disp_draw_buf_t* variable to initialize
- **buf1** -- A buffer to be used by LVGL to draw the image. Always has to specified and can't be NULL. Can be an array allocated by the user. E.g. `static lv_color_t disp_buf1[1024 * 10]` Or a memory address e.g. in external SRAM
- **buf2** -- Optionally specify a second buffer to make image rendering and image flushing (sending to the display) parallel. In the `disp_drv->flush` you should use DMA or similar hardware to send the image to the display in the background. It lets LVGL to render next frame into the other buffer while previous is being sent. Set to NULL if unused.
- **size_in_px_cnt** -- size of the buf1 and buf2 in pixel count.

lv_disp_t ***lv_disp_drv_register**(*lv_disp_drv_t* *driver)

Register an initialized display driver. Automatically set the first display as active.

参数 **driver** -- pointer to an initialized '`lv_disp_drv_t`' variable. Only its pointer is saved!

返回 pointer to the new display or NULL on error

void lv_disp_drv_update(lv_disp_t *disp, lv_disp_drv_t *new_drv)
Update the driver in run time.

参数

- **disp** -- pointer to a display. (return value of `lv_disp_drv_register()`)
- **new_drv** -- pointer to the new driver

void lv_disp_remove(lv_disp_t *disp)
Remove a display

参数 disp -- pointer to display

void lv_disp_set_default(lv_disp_t *disp)
Set a default display. The new screens will be created on it by default.

参数 disp -- pointer to a display

lv_disp_t *lv_disp_get_default(void)
Get the default display

返回 pointer to the default display

lv_coord_t lv_disp_get_hor_res(lv_disp_t *disp)
Get the horizontal resolution of a display

参数 disp -- pointer to a display (NULL to use the default display)

返回 the horizontal resolution of the display

lv_coord_t lv_disp_get_ver_res(lv_disp_t *disp)
Get the vertical resolution of a display

参数 disp -- pointer to a display (NULL to use the default display)

返回 the vertical resolution of the display

bool lv_disp_get_antialiasing(lv_disp_t *disp)
Get if anti-aliasing is enabled for a display or not

参数 disp -- pointer to a display (NULL to use the default display)

返回 true: anti-aliasing is enabled; false: disabled

lv_coord_t lv_disp_get_dpi(const lv_disp_t *disp)
Get the DPI of the display

参数 disp -- pointer to a display (NULL to use the default display)

返回 dpi of the display

void lv_disp_set_rotation(lv_disp_t *disp, lv_disp_rot_t rotation)
Set the rotation of this display.

参数

- **disp** -- pointer to a display (NULL to use the default display)
- **rotation** -- rotation angle

lv_disp_rot_t lv_disp_get_rotation(lv_disp_t *disp)
Get the current rotation of this display.

参数 disp -- pointer to a display (NULL to use the default display)

返回 rotation angle

`lv_disp_t *lv_disp_get_next(lv_disp_t *disp)`
Get the next display.

参数 **disp** -- pointer to the current display. NULL to initialize.

返回 the next display or NULL if no more. Give the first display when the parameter is NULL

`lv_disp_draw_buf_t *lv_disp_get_draw_buf(lv_disp_t *disp)`
Get the internal buffer of a display

参数 **disp** -- pointer to a display

返回 pointer to the internal buffers

struct **lv_disp_draw_buf_t**

#include <lv_hal_disp.h> Structure for holding display buffer information.

Public Members

void *buf1

First display buffer.

void *buf2

Second display buffer.

void *buf_act

uint32_t size

lv_area_t area

int flushing

int flushing_last

uint32_t last_area

uint32_t last_part

struct **lv_disp_drv_t**

#include <lv_hal_disp.h> Display Driver structure to be registered by HAL. Only its pointer will be saved in `lv_disp_t` so it should be declared as `static lv_disp_drv_t my_drv` or allocated dynamically.

Public Members

lv_coord_t hor_res

Horizontal resolution.

lv_coord_t ver_res

Vertical resolution.

lv_disp_draw_buf_t *draw_buf

Pointer to a buffer initialized with `lv_disp_draw_buf_init()`. LVGL will use this buffer(s) to draw the screens contents

uint32_t full_refresh

1: Always make the whole screen redrawn

uint32_t sw_rotate

1: use software rotation (slower)

uint32_t antialiasing

1: anti-aliasing is enabled on this display.

uint32_t rotated

1: turn the display by 90 degree.

警告: Does not update coordinates for you!

uint32_t screen_transp**uint32_t dpi**

Handle if the screen doesn't have a solid (opa == LV_OPA_COVER) background. Use only if required because it's slower.

void (*flush_cb)(struct *lv_disp_drv_t* *disp_drv, const lv_area_t *area, lv_color_t *color_p)

DPI (dot per inch) of the display. Default value is LV_DPI_DEF. MANDATORY: Write the internal buffer (draw_buf) to the display. 'lv_disp_flush_ready()' has to be called when finished

void (*rounder_cb)(struct *lv_disp_drv_t* *disp_drv, lv_area_t *area)

OPTIONAL: Extend the invalidated areas to match with the display drivers requirements E.g. round y to, 8, 16 ..) on a monochrome display

void (*set_px_cb)(struct *lv_disp_drv_t* *disp_drv, uint8_t *buf, lv_coord_t buf_w, lv_coord_t x, lv_coord_t y, lv_color_t color, lv_opa_t opa)

OPTIONAL: Set a pixel in a buffer according to the special requirements of the display Can be used for color format not supported in LittlevGL. E.g. 2 bit -> 4 gray scales

注解: Much slower then drawing with supported color formats.

void (*monitor_cb)(struct *lv_disp_drv_t* *disp_drv, uint32_t time, uint32_t px)

OPTIONAL: Called after every refresh cycle to tell the rendering and flushing time + the number of flushed pixels

void (*wait_cb)(struct *lv_disp_drv_t* *disp_drv)

OPTIONAL: Called periodically while lvgl waits for operation to be completed. For example flushing or GPU User can execute very simple tasks here or yield the task

void (*clean_dcache_cb)(struct *lv_disp_drv_t* *disp_drv)

OPTIONAL: Called when lvgl needs any CPU cache that affects rendering to be cleaned

void (*gpu_wait_cb)(struct *lv_disp_drv_t* *disp_drv)

OPTIONAL: called to wait while the gpu is working

`void (*drv_update_cb)(struct _lv_disp_drv_t *disp_drv)`
OPTIONAL: called when driver parameters are updated

`void (*gpu_fill_cb)(struct _lv_disp_drv_t *disp_drv, lv_color_t *dest_buf, lv_coord_t dest_width, const lv_area_t *fill_area, lv_color_t color)`
OPTIONAL: Fill a memory with a color (GPU only)

lv_color_t **color_chroma_key**

On CHROMA_KEYED images this color will be transparent. `LV_COLOR_CHROMA_KEY` by default.
(lv_conf.h)

`void *user_data`
Custom display driver user data

`struct _lv_disp_t`
`#include <lv_hal_disp.h>` Display structure.

注解: `lv_disp_drv_t` should be the first member of the structure.

Public Members

`lv_disp_drv_t *driver`

< Driver to the display A timer which periodically checks the dirty areas and refreshes them

`lv_timer_t *refr_timer`

The theme assigned to the screen

`struct _lv_theme_t *theme`

`struct _lv_obj_t **screens`

Screens of the display Array of screen objects.

`struct _lv_obj_t *act_scr`

Currently active screen on this display

`struct _lv_obj_t *prev_scr`

Previous screen. Used during screen animations

`struct _lv_obj_t *scr_to_load`

The screen prepared to load in `lv_scr_load_anim`

`struct _lv_obj_t *top_layer`

See [lv_disp_get_layer_top](#)

`struct _lv_obj_t *sys_layer`

See [lv_disp_get_layer_sys](#)

`uint32_t screen_cnt`

uint8_t del_prev

1: Automatically delete the previous screen when the screen load animation is ready

lv_opa_t bg_opa

Opacity of the background color or wallpaper

lv_color_t bg_color

Default display color when screens are transparent

const void *bg_img

An image source to display as wallpaper

lv_area_t inv_areas[LV_INV_BUF_SIZE]

Invalidated (marked to redraw) areas

uint8_t inv_area_joined[LV_INV_BUF_SIZE]**uint16_t inv_p****uint32_t last_activity_time**

Last time when there was activity on this display

4.3 Input device interface (输入设备接口)

4.3.1 Types of input devices (输入设备的类型)

To register an input device an `lv_indev_drv_t` variable has to be initialized:

要注册输入设备，必须初始化一个 `lv_indev_drv_t` 变量：

```
lv_indev_drv_t indev_drv;
lv_indev_drv_init(&indev_drv);           /*Basic initialization*/
indev_drv.type = ...                   /*See below.*/
indev_drv.read_cb = ...                /*See below.*/
/*Register the driver in LVGL and save the created input device object*/
lv_indev_t * my_indev = lv_indev_drv_register(&indev_drv);
```

`type` 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

`read_cb` is a function pointer which will be called periodically to report the current state of an input device.

Visit [Input devices](#) to learn more about input devices in general.

`type` 可以是

- `LV_INDEV_TYPE_POINTER` 触摸板或鼠标
- `LV_INDEV_TYPE_KEYPAD` 键盘或小键盘
- `LV_INDEV_TYPE_ENCODER` 编码器，带有左/右转和推动选项

- LV_INDEV_TYPE_BUTTON 外部按钮虚拟按下屏幕
`read_cb` 是一个函数指针，它将被定期调用以报告输入设备的当前状态。
[访问输入设备](#) 以了解有关输入设备的更多信息。

Touchpad, mouse or any pointer (触摸板、鼠标或任何指针)

Input devices that can click points on the screen belong to this category.

可以点击屏幕上的点的输入设备属于这一类。

```
indev_drv.type = LV_INDEV_TYPE_POINTER;
indev_drv.read_cb = my_input_read;

...

void my_input_read(lv_indev_drv_t * drv, 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;
    }
}
```

To set a mouse cursor use `lv_indev_set_cursor(my_indev, &img_cursor)`. (`my_indev` is the return value of `lv_indev_drv_register`)

要设置鼠标光标，请使用 `lv_indev_set_cursor(my_indev, &img_cursor)`。(`my_indev` 是 `lv_indev_drv_register` 的返回值)

Keypad or keyboard (键盘或键盘)

Full keyboards with all the letters or simple keypads with a few navigation buttons belong here.

To use a keyboard/keypad:

- Register a `read_cb` function with `LV_INDEV_TYPE_KEYPAD` type.
- An object group has to be created: `lv_group_t * g = lv_group_create()` and objects have to be added to it with `lv_group_add_obj(g, obj)`
- The created group has to be assigned to an input device: `lv_indev_set_group(my_indev, g)` (`my_indev` is the return value of `lv_indev_drv_register`)
- Use `LV_KEY_...` to navigate among the objects in the group. See `lv_core/lv_group.h` for the available keys.

带有所有字母的全键盘或带有几个导航按钮的简单键盘都属于这里。

要使用键盘/小键盘：

- 注册一个带有 `LV_INDEV_TYPE_KEYPAD` 类型的 `read_cb` 函数。
- 必须创建一个对象组: `lv_group_t * g = lv_group_create()` 并且对象必须使用 `lv_group_add_obj(g, obj)` 添加到其中

- 创建的组必须分配给输入设备: `lv_indev_set_group(my_indev, g)` (`my_indev` 是 `lv_indev_drv_register` 的返回值)
- 使用 `LV_KEY_...` 在组中的对象之间导航。有关可用密钥, 请参阅 “`lv_core/lv_group.h`”。

```
indev_drv.type = LV_INDEV_TYPE_KEYPAD;
indev_drv.read_cb = keyboard_read;

...

void keyboard_read(lv_indev_drv_t * drv, lv_indev_data_t*data){
    data->key = last_key(); /*Get the last pressed or released key*/

    if(key_pressed()) data->state = LV_INDEV_STATE_PRESSED;
    else data->state = LV_INDEV_STATE_RELEASED;
}
```

Encoder (编码器)

With an encoder you can do 4 things:

1. Press its button
2. Long-press its button
3. Turn left
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 object will go to edit mode whereby turning the encoder you can navigate inside the object.
- To leave edit mode press long the button.

To use an *Encoder* (similarly to the *Keypads*) the objects should be added to groups.

使用编码器, 您可以做 4 件事:

1. 按下它的按钮
2. 长按它的按钮
3. 左转
4. 右转

简而言之, 编码器输入设备的工作方式如下:

- 通过转动编码器, 您可以专注于下一个/上一个对象。
- 当您在一个简单的对象 (如按钮) 上按下编码器时, 它将被点击。
- 如果您按下复杂对象 (如列表、消息框等) 上的编码器, 该对象将进入编辑模式, 从而转动编码器您可以在对象内部导航。
- 要退出编辑模式, 请长按按钮。

要使用 *Encoder* (类似于 *Keypads*), 应将对象添加到组中。

```

indev_drv.type = LV_INDEV_TYPE_ENCODER;
indev_drv.read_cb = encoder_read;

...
void encoder_read(lv_indev_drv_t * drv, 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;
}

```

Using buttons with Encoder logic (使用带有编码器逻辑的按钮)

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 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 encoder click with period specified in `indev_drv.long_press_rep_time`.

除了标准编码器行为之外，您还可以利用其逻辑来使用按钮导航（聚焦）和编辑小部件。如果您只有几个按钮可用，或者您想使用除编码轮之外的其他按钮，这将特别方便。

您需要有 3 个按钮可用：

- LV_KEY_ENTER 将模拟按下或按下编码器按钮
- LV_KEY_LEFT 将模拟向左转动编码器
- LV_KEY_RIGHT 将模拟向右旋转编码器
- 其他键将传递给聚焦的小部件

如果您按住这些键，它将模拟编码器点击，并在 `indev_drv.long_press_rep_time` 中指定周期。

```

indev_drv.type = LV_INDEV_TYPE_ENCODER;
indev_drv.read_cb = encoder_with_keys_read;

...
void encoder_with_keys_read(lv_indev_drv_t * drv, 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();
    }
}

```

Button (按钮)

Buttons mean external "hardware" buttons next to the screen which are assigned to specific coordinates of the screen. If a button is pressed it will simulate the pressing on the assigned coordinate. (Similarly to a touchpad)

To assign 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}, ... }`

按钮表示屏幕旁边的外部“硬件”按钮，这些按钮分配给屏幕的特定坐标。如果按下按钮，它将模拟按下指定坐标。(类似于触摸板)

要将按钮分配给坐标，请使用 `lv_indev_set_button_points(my_indev, points_array)`。`points_array` 应该看起来像 `const lv_point_t points_array[] = { {12, 30}, {60, 90}, ... }`

重要: The `points_array` can't go out of scope. Either declare it as a global variable or as a static variable inside a function.

```
indev_drv.type = LV_INDEV_TYPE_BUTTON;
indev_drv.read_cb = button_read;

...
void button_read(lv_indev_drv_t *drv, 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 = last_btn; /*Save the last button*/
}
```

4.3.2 Other features (其它功能)

Parameters (参数)

The default value of the following parameters can changed in `lv_indev_drv_t`:

- `scroll_limit` Number of pixels to slide before actually scrolling the object.
- `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_rep_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 `lv_timer_...()` functions. `LV_INDEV_DEF_READ_PERIOD` in `lv_conf.h` sets the default read period.

以下参数的默认值可以在 `lv_indev_drv_t` 中更改:

- `scroll_limit` 在实际滚动对象之前要滑动的像素数。
- `scroll_throw` 滚动投掷（动量）减慢 [%]。更大的值意味着更快的减速。
- `long_press_time` 按下发送 `LV_EVENT_LONG_PRESSED` 的时间（以毫秒为单位）
- `long_press_rep_time` 发送 `LV_EVENT_LONG_PRESSED_REPEAT` 的间隔（以毫秒为单位）
- `read_timer` 指向读取输入设备的 `lv_timer` 的指针。它的参数可以通过 `lv_timer_...()` 函数改变。`lv_conf.h` 中的 `LV_INDEV_DEF_READ_PERIOD` 设置默认读取周期。

Feedback (回调处理)

Besides `read_cb` a `feedback_cb` callback can be also specified in `lv_indev_drv_t`. `feedback_cb` is called when any type of event is sent by the input devices (independently from its type). This allows generating feedback for the user, e.g. to play a sound on `LV_EVENT_CLICKED`.

除了 `read_cb` 一个 `feedback_cb` 回调也可以在 `lv_indev_drv_t` 中指定。`feedback_cb` 在输入设备发送任何类型的事件时被调用（与其类型无关）。这允许为用户生成反馈，例如在“`LV_EVENT_CLICKED`”上播放声音。

Associating with a display (与显示器关联)

Every input device is associated with a display. By default, a new input device is added to the lastly created or the explicitly selected (using `lv_disp_set_default()`) display. The associated display is stored and can be changed in `disp` field of the driver.

每个输入设备都与一个显示器相关联。默认情况下，一个新的输入设备被添加到最后创建的或明确选择的（使用 `lv_disp_set_default()`）显示。相关的显示被存储并可在驱动程序的“`disp`”字段中更改。

Buffered reading (缓冲读取)

By default LVGL calls `read_cb` periodically. This way 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 set the buffered data instead of reading the input device. You can set the `data->continue_reading` flag to tell that LVGL there is more data to read and it should call the `read_cb` again.

默认情况下，LVGL 会定期调用 `read_cb`。这样就有可能错过一些用户手势。

为了解决这个问题，你可以为你的输入设备编写一个事件驱动的驱动程序来缓冲测量数据。在 `read_cb` 中，您可以设置缓冲数据而不是读取输入设备。你可以设置 `data->continue_reading` 标志来告诉 LVGL 有更多的数据要读取，它应该再次调用 `read_cb`。

4.3.3 Further reading (深入学习)

- `lv_port_indev_template.c` for a template for your own driver.
- `INdev features` to learn more about higher level input device features.
- `lv_port_indev_template.c` 用于您自己的驱动程序的模板。
- `INdev features` 以了解有关更高级别输入设备功能的更多信息。

4.3.4 API

@description Input Device HAL interface layer header file

Typedefs

typedef struct *lv_indev_drv_t* **lv_indev_drv_t**
Initialized by the user and registered by 'lv_indev_add()'

typedef struct *lv_indev_proc_t* **lv_indev_proc_t**
Run time data of input devices Internally used by the library, you should not need to touch it.

typedef struct *lv_indev_t* **lv_indev_t**
The main input device descriptor with driver, runtime data ('proc') and some additional information

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**

Functions

`void lv_indev_drv_init(lv_indev_drv_t *driver)`

Initialize an input device driver with default values. It is used to surely have known values in the fields and not memory junk. After it you can set the fields.

参数 **driver** -- pointer to driver variable to initialize

`lv_indev_t *lv_indev_drv_register(lv_indev_drv_t *driver)`

Register an initialized input device driver.

参数 **driver** -- pointer to an initialized 'lv_indev_drv_t' variable (can be local variable)

返回 pointer to the new input device or NULL on error

`void lv_indev_drv_update(lv_indev_t *indev, lv_indev_drv_t *new_drv)`

Update the driver in run time.

参数

- **indev** -- pointer to a input device. (return value of `lv_indev_drv_register`)
- **new_drv** -- pointer to the new driver

`lv_indev_t *lv_indev_get_next(lv_indev_t *indev)`

Get the next input device.

参数 **indev** -- pointer to the current input device. NULL to initialize.

返回 the next input device or NULL if no more. Give the first input device when the parameter is NULL

`void _lv_indev_read(lv_indev_t *indev, lv_indev_data_t *data)`

Read data from an input device.

参数

- **indev** -- pointer to an input device
- **data** -- input device will write its data here

struct `lv_indev_data_t`

#include <lv_hal_indev.h> Data structure passed to an input driver to fill

Public Members

`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

lv_indev_state_t state
LV_INDEV_STATE_REL or LV_INDEV_STATE_PR

bool continue_reading
Call the read callback until it's set to true

struct _lv_indev_drv_t
#include <lv_hal_indev.h> Initialized by the user and registered by 'lv_indev_add()'

Public Members

lv_indev_type_t type
< Input device type Function pointer to read input device data.

void (*read_cb)(struct _lv_indev_drv_t *indev_drv, lv_indev_data_t *data)

void (*feedback_cb)(struct _lv_indev_drv_t*, uint8_t)

Called when an action happened on the input device. The second parameter is the event from **lv_event_t**

void *user_data

struct _lv_disp_t *disp

< Pointer to the assigned display 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 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

struct _lv_indev_proc_t

#include <lv_hal_indev.h> Run time data of input devices Internally used by the library, you should not need to touch it.

Public Members

`lv_indev_state_t state`

Current state of the input device.

`uint8_t long_pr_sent`

`uint8_t reset_query`

`uint8_t disabled`

`uint8_t wait_until_release`

`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 scroll_sum`

`lv_point_t scroll_throw_vect`

`lv_point_t scroll_throw_vect_ori`

`struct _lv_obj_t *act_obj`

`struct _lv_obj_t *last_obj`

`struct _lv_obj_t *scroll_obj`

`struct _lv_obj_t *last_pressed`

`lv_area_t scroll_area`

`lv_point_t gesture_sum`

`lv_dir_t scroll_dir`

`lv_dir_t gesture_dir`

`uint8_t gesture_sent`

`struct _lv_indev_proc_t::[anonymous]::[anonymous] pointer`

`lv_indev_state_t last_state`

`uint32_t last_key`

`struct _lv_indev_proc_t::[anonymous]::[anonymous] keypad`

`union _lv_indev_proc_t::[anonymous] types`

`uint32_t pr_timestamp`

Pressed time stamp

```
uint32_t longpr_rep_timestamp
```

Long press repeat time stamp

```
struct _lv_indev_t
```

#include <lv_hal_indev.h> The main input device descriptor with driver, runtime data ('proc') and some additional information

Public Members

```
lv_indev_drv_t *driver
```

```
lv_indev_proc_t proc
```

```
struct _lv_obj_t *cursor
```

Cursor for LV_INPUT_TYPE_POINTER

```
struct _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

4.4 Tick interface (心跳接口)

LVGL needs a system tick to know elapsed time for animations and other tasks.

You need to call the `lv_tick_inc(tick_period)` function periodically and provide the call period in milliseconds. For example, `lv_tick_inc(1)` when calling every millisecond.

`lv_tick_inc` should be called in a higher priority routine than `lv_task_handler()` (e.g. in an interrupt) to precisely know the elapsed milliseconds even if the execution of `lv_task_handler` takes more time.

With FreeRTOS `lv_tick_inc` can be called in `vApplicationTickHook`.

On Linux based operating system (e.g. on Raspberry Pi) `lv_tick_inc` can be called in a thread like below:

LVGL 需要一个系统滴答来了解动画和其他任务所用的时间。

您需要定期调用 `lv_tick_inc(tick_period)` 函数并提供以毫秒为单位的调用周期。例如, `lv_tick_inc(1)` 每毫秒调用一次。

`lv_tick_inc` 应该在比 `lv_task_handler()` 更高优先级的例程中调用 (例如在中断中), 以精确知道经过的毫秒数, 即使 `lv_task_handler` 的执行需要更多时间。

使用 FreeRTOS, 可以在 `vApplicationTickHook` 中调用 `lv_tick_inc`。

在基于 Linux 的操作系统 (例如在 Raspberry Pi 上) 可以在如下线程中调用 `lv_tick_inc`:

```
void * tick_thread (void *args)
{
    while(1) {
        usleep(5*1000); /*Sleep for 5 millisecond*/
        lv_tick_inc(5); /*Tell LVGL that 5 milliseconds were elapsed*/
    }
}
```

4.4.1 API

Provide access to the system tick with 1 millisecond resolution

Functions

`uint32_t lv_tick_get(void)`

Get the elapsed milliseconds since start up

返回 the elapsed milliseconds

`uint32_t lv_tick_elaps(uint32_t prev_tick)`

Get the elapsed milliseconds since a previous time stamp

参数 `prev_tick` -- a previous time stamp (return value of `lv_tick_get()`)

返回 the elapsed milliseconds since 'prev_tick'

4.5 Task Handler (任务处理器)

To handle the tasks of LVGL you need to call `lv_timer_handler()` periodically in one of the following:

- `while(1)` of `main()` function
- timer interrupt periodically (lower priority than `lv_tick_inc()`)
- an OS task periodically

The timing is not critical but it should be about 5 milliseconds to keep the system responsive.

Example:

要处理 LVGL 的任务，您需要以下列方式之一定期调用 `lv_timer_handler()`:

- `main()` 函数的 `while(1)`
- 定时器定期中断 (比 `lv_tick_inc()` 优先级低)
- 定期执行操作系统任务

时间并不重要，但它应该是大约 5 毫秒以保持系统响应。

示例:

```
while(1) {
    lv_timer_handler();
    my_delay_ms(5);
}
```

To learn more about timers visit the [Timer](#) section.

要了解有关计时器的更多信息，请访问[Timer](#) 部分。

4.6 Sleep management (睡眠管理)

The MCU can go to sleep when no user input happens. In this case, the main `while(1)` should look like this:

当没有用户输入发生时，MCU 可以进入睡眠状态。在这种情况下，主要的 `while(1)` 应该是这样的：

```
while(1) {
    /*Normal operation (no sleep) in < 1 sec inactivity*/
    if(lv_disp_get_inactive_time(NULL) < 1000) {
        lv_task_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);
}
```

You should also add the below lines to your input device read function to signal a wake-up (press, touch or click etc.) happened:

您还应该将以下几行添加到您的输入设备读取功能中，以表示发生了唤醒（按下、触摸或点击等）：

```
lv_tick_inc(LV_DISP_DEF_REFR_PERIOD); /*Force task execution on wake-up*/
timer_start();                         /*Restart the timer where lv_tick_inc() is
→called*/
lv_task_handler();                     /*Call `lv_task_handler()` manually to process
→the wake-up event*/
```

In addition to `lv_disp_get_inactive_time()` you can check `lv_anim_count_running()` to see if all animations have finished.

除了 `lv_disp_get_inactive_time()` 之外，你还可以检查 `lv_anim_count_running()` 以查看是否所有动画都已完成。

4.7 Operating system and interrupts (操作系统和中断)

LVGL is **not thread-safe** by default.

However, in the following conditions it's valid to call LVGL related functions:

- In `events`. Learn more in [Events](#).
- In `lv_timer`. Learn more in [Timers](#).

默认情况下，LVGL 非线程安全。

但是，在以下情况下调用 LVGL 相关函数是有效的：

- 在事件中。在[事件](#)中了解更多信息。
- 在 `lv_timer` 中。在[计时器](#)中了解更多信息。

4.7.1 Tasks and threads (任务和线程)

If you need to use real tasks or threads, you need a mutex which should be invoked before the call of `lv_timer_handler` and released after it. Also, you have to use the same mutex in other tasks and threads around every LVGL (`lv_...`) related function call and code. This way you can use LVGL in a real multitasking environment. Just make use of a mutex to avoid the concurrent calling of LVGL functions.

如果你需要使用真正的任务或线程，你需要一个互斥锁，它应该在调用 `lv_timer_handler` 之前被调用并在它之后释放。此外，您必须在每个 LVGL (`lv_...`) 相关函数调用和代码周围的其他任务和线程中使用相同的互斥锁。这样你就可以在真正的多任务环境中使用 LVGL。只需使用互斥锁来避免并发调用 LVGL 函数。

4.7.2 Interrupts (中断)

Try to avoid calling LVGL functions from interrupt handlers (except `lv_tick_inc()` and `lv_disp_flush_ready()`). But if you need to do this you have to disable the interrupt which uses LVGL functions while `lv_timer_handler` is running. It's a better approach to set a flag or some value and periodically check it in an `lv_timer`.

尽量避免从中断处理程序调用 LVGL 函数（除了 `lv_tick_inc()` 和 `lv_disp_flush_ready()`）。但是如果你需要这样做，你必须在 `lv_timer_handler` 运行时禁用使用 LVGL 函数的中断。设置一个标志或某个值并在 `lv_timer` 中定期检查它是一种更好的方法。

4.8 Logging (日志)

LVGL has built-in *Log* module to inform the user about what is happening in the library.

LVGL 有内置的 *Log* 模块来通知用户库中发生的事情。

4.8.1 Log level (日记等级)

To enable logging, set `LV_USE_LOG 1` 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 too. E.g. if you `LV_LOG_LEVEL_WARN`, errors will be also logged.

要启用日志记录，请在“`lv_conf.h`”中设置“`LV_USE_LOG 1`”并将“`LV_LOG_LEVEL`”设置为以下值之一：

- `LV_LOG_LEVEL_TRACE` 大量日志提供详细信息
- `LV_LOG_LEVEL_INFO` 记录重要事件
- `LV_LOG_LEVEL_WARN` 记录是否发生了不想要的事情但没有引起问题
- `LV_LOG_LEVEL_ERROR` 只有关键问题，系统可能会失败
- `LV_LOG_LEVEL_USER` 仅用户消息

- `LV_LOG_LEVEL_NONE` 不记录任何内容

级别高于设置的日志级别的事件也将被记录。例如。如果你 `LV_LOG_LEVEL_WARN`, 错误也会被记录。

4.8.2 Printing logs (打印日志)

Logging with printf (使用 printf 记录)

If your system supports `printf`, you just need to enable `LV_LOG_PRINTF` in `lv_conf.h` to send the logs with `printf`.

如果你的系统支持 `printf`, 你只需要在 `lv_conf.h` 中启用 `LV_LOG_PRINTF` 就可以发送带有 `printf` 的日志。

Custom log function (自定义日志功能)

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()`.

For example:

如果你不能使用 `printf` 或者想使用自定义函数来记录日志, 你可以使用 `lv_log_register_print_cb()` 注册一个“记录器”回调。

例如:

```
void my_log_cb(const char * buf)
{
    serial_send(buf, strlen(buf));
}

...
lv_log_register_print_cb(my_log_cb);
```

4.8.3 Add logs (添加日志)

You can also use the log module via the `LV_LOG_TRACE/INFO/WARN/ERROR/USER(text)` functions.

您还可以通过 `LV_LOG_TRACE/INFO/WARN/ERROR/USER(text)` 函数使用日志模块。

CHAPTER 5

Overview (概览)

5.1 Objects (对象)

In LVGL the **basic building blocks** of a user interface are the objects, also called *Widgets*. For example a *Button*, *Label*, *Image*, *List*, *Chart* or *Text area*.

You can see all the *Object types* here.

All objects are referenced using an `lv_obj_t` pointer as a handle. This pointer can later be used to set or get the attributes of the object.

在 LVGL 中，用户界面的基本构建块是对象，也称为 *Widgets*。例如 *Button*、*Label*、*Image*、*List*，图表或文本区域。

您可以在此处查看所有对象类型。

所有对象都使用 `lv_obj_t` 指针作为句柄进行引用。此指针稍后可用于设置或获取对象的属性。

5.1.1 Attributes (属性)

Basic attributes (基本属性)

All object types share some basic attributes:

- Position
- Size
- Parent
- Styles
- Event handlers
- Etc

You can set/get these attributes with `lv_obj_set_...` and `lv_obj_get_...` functions. For example:

所有对象类型共享一些基本属性：

- 位置
- 尺寸
- 家长
- 样式
- 事件处理程序
- 等等

您可以使用 `lv_obj_set_...` 和 `lv_obj_get_...` 函数设置/获取这些属性。例如：

```
/*Set basic object 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*/
```

To see all the available functions visit the [Base object's documentation](#).

要查看所有可用函数，请访问[Base object's documentation](#)。

Specific attributes (特定属性)

The object types have special attributes too. For example, a slider has

- Minimum and maximum values
- Current value

For these special attributes, every object type may have unique API functions. For example for a slider:

对象类型也有特殊的属性。例如，一个滑块有

- 最小值和最大值
- 当前值

对于这些特殊的属性，每个对象类型都可能有唯一的 API 函数。例如对于滑块：

```
/*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)*/
```

The API of the widgets is described in their [Documentation](#) but you can also check the respective header files (e.g. `widgets/lv_slider.h`)

小部件的 API 在它们的[文档](#)中有描述，但您也可以检查相应的头文件（例如 `widgets/lv_slider.h`）

5.1.2 Working mechanisms (工作机制)

Parent-child structure (父子结构)

A parent object can be considered as the container of its children. Every object has exactly one parent object (except screens), but a parent can have any number of children. There is no limitation for the type of the parent but, there are typical parent (e.g. button) and typical child (e.g. label) objects.

父对象可以被视为其子对象的容器。每个对象只有一个父对象（屏幕除外），但一个父对象可以有任意数量的子对象。父对象的类型没有限制，但是有典型的父对象（例如按钮）和典型的子对象（例如标签）。

Moving together (一起移动)

If the position of the parent changes the children will move with the parent. Therefore all positions are relative to the parent.

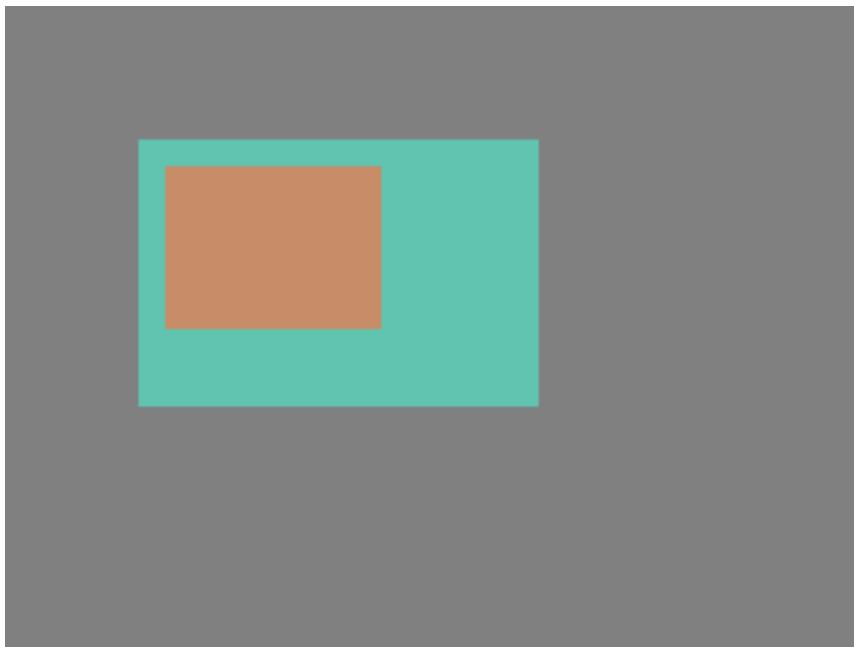
如果父节点的位置发生变化，子节点将与父节点一起移动。因此，所有位置都相对于父级。



```
lv_obj_t * parent = lv_obj_create(lv_scr_act()); /*Create a parent object on the
→current screen*/
lv_obj_set_size(parent, 100, 80); /*Set the size of the
→parent*/
lv_obj_t * obj1 = lv_obj_create(parent); /*Create an object on the
→previously created parent object*/
lv_obj_set_pos(obj1, 10, 10); /*Set the position of the
→new object*/
```

Modify the position of the parent:

修改父级的位置：



```
lv_obj_set_pos(parent, 50, 50);           /*Move the parent. The child will move with it.  
→*/
```

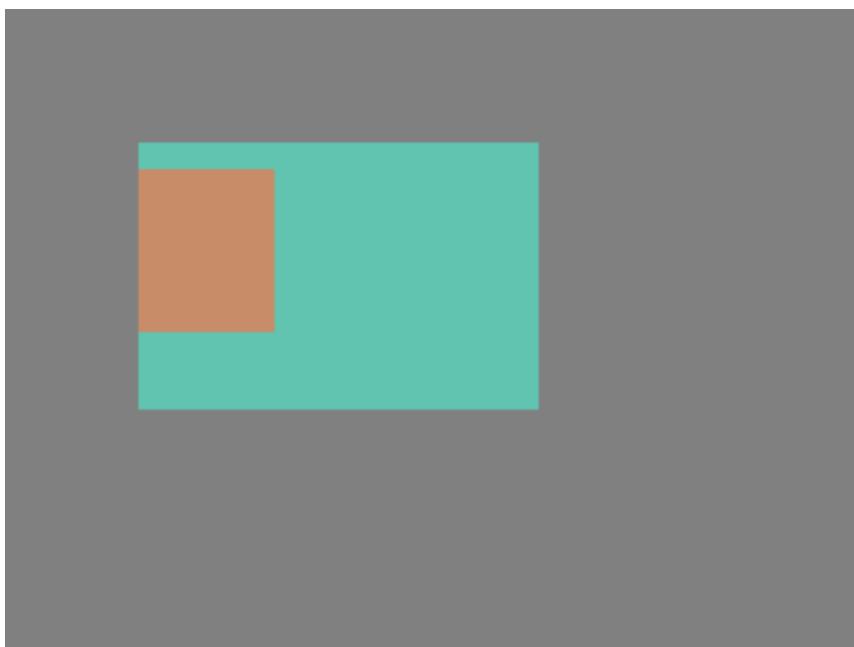
(For simplicity the adjusting of colors of the objects is not shown in the example.)

(为简单起见，示例中未显示对象颜色的调整。)

Visibility only on the parent (仅在父对象上可见)

If a child is partially or fully out of its parent then the parts outside will not be visible.

如果孩子部分或完全脱离其父对象，则外部部分将不可见。



```
lv_obj_set_x(obj1, -30);           /*Move the child a little bit off the parent*/
```

Create and delete objects (创建和删除对象)

In LVGL objects can be created and deleted dynamically in run time. It means only the currently created (existing) objects consume RAM.

This allows for the creation of a screen just when a button is clicked to open it, and for deletion of screens when a new screen is loaded.

UIs can be created based on the current environment of the device. For example one can create meters, charts, bars and sliders based on the currently attached sensors.

Every widget has its own **create** function with a prototype like this:

在 LVGL 中，可以在运行时动态创建和删除对象。这意味着只有当前创建的（现有）对象消耗 RAM。

这允许仅在单击按钮打开屏幕时创建屏幕，并在加载新屏幕时删除屏幕。

可以根据设备的当前环境创建 UI。例如，可以根据当前连接的传感器创建仪表、图表、条形图和滑块。

每个小部件都有自己的 **create** 函数，原型如下：

```
lv_obj_t * lv_<widget>_create(lv_obj_t * parent, <other parameters if any>);
```

In most of the cases the create functions have only a *parent* parameter that tells on which object create the new widget.

The return value is a pointer to the created object with *lv_obj_t* * type.

There is a common **delete** function for all object types. It deletes the object and all of its children.

在大多数情况下，**create** 函数只有一个 *parent* 参数，它告诉在哪个对象上创建新小部件。

返回值是一个指向创建对象的指针，类型为 *lv_obj_t* *。

所有对象类型都有一个通用的 **delete** 函数。它删除对象及其所有子对象。

```
void lv_obj_del(lv_obj_t * obj);
```

lv_obj_del will delete the object immediately. If for any reason you can't delete the object immediately you can use *lv_obj_del_async(obj)* that will perform the deletion on the next call of *lv_timer_handler()*. This is useful e.g. if you want to delete the parent of an object in the child's *LV_EVENT_DELETE* handler.

You can remove all the children of an object (but not the object itself) using *lv_obj_clean(obj)*.

You can use *lv_obj_del_delayed(obj, 1000)* to delete an object after some time. The delay is expressed in milliseconds.

lv_obj_del 将立即删除对象。如果由于任何原因你不能立即删除对象，你可以使用 *lv_obj_del_async(obj)* 它将在下一次调用 *lv_timer_handler()* 时执行删除。这很有用，例如如果您想在子对象的 *LV_EVENT_DELETE* 处理程序中删除对象的父对象。您可以使用 *lv_obj_clean(obj)* 删除对象的所有子项（但不是对象本身）。一段时间后，您可以使用 *lv_obj_del_delayed(obj, 1000)* 来删除对象。延迟以毫秒表示。

5.1.3 Screens (屏幕)

Create screens (创建屏幕)

The screens are special objects which have no parent object. So they can be created like:

屏幕是没有父对象的特殊对象。所以它们可以像这样创建：

```
lv_obj_t * scr1 = lv_obj_create(NULL);
```

Screens can be created with any object type. For example, a *Base object* or an image to make a wallpaper.

可以使用任何对象类型创建画面。例如，基础对象 或用于制作壁纸的图像。

Get the active screen (获取活动屏幕)

There is always an active screen on each display. By default, the library creates and loads a "Base object" as a screen for each display.

To get the currently active screen use the `lv_scr_act()` function.

每个显示器上总是有一个活动屏幕。默认情况下，库创建并加载一个“基础对象”作为每个显示的屏幕。

要获取当前活动的屏幕，请使用 `lv_scr_act()` 函数。

Load screens (加载屏幕)

To load a new screen, use `lv_scr_load(scr1)`.

请使用 `lv_scr_load(scr1)` 加载你要加载的屏幕。

Layers (层)

There are two automatically generated layers:

- top layer
- system layer

They are independent of the screens and they will be shown on every screen. The *top layer* is above every object on the screen and the *system layer* is above the *top layer* too. You can add any pop-up windows to the *top layer* freely. But, the *system layer* is restricted to system-level things (e.g. mouse cursor will be placed here in `lv_indev_set_cursor()`).

The `lv_layer_top()` and `lv_layer_sys()` functions return pointers to the top and system layers respectively.

Read the [Layer overview](#) section to learn more about layers.

有两个自动生成的层：

- 顶层
- 系统层

它们独立于屏幕，将显示在每个屏幕上。顶层位于屏幕上的每个对象之上，系统层也位于顶层之上。您可以自由地将任何弹出窗口添加到顶层。但是，系统层仅限于系统级事物（例如，鼠标光标将放置在“`lv_indev_set_cursor()`”中）。

`lv_layer_top()` 和 `lv_layer_sys()` 函数分别返回指向顶层和系统层的指针。

阅读[图层概述](#)部分以了解有关图层的更多信息。

Load screen with animation (用动画加载屏幕)

A new screen can be loaded with animation too using `lv_scr_load_anim(scr, transition_type, time, delay, auto_del)`. The following transition types exist:

- `LV_SCR_LOAD_ANIM_NONE`: switch immediately after `delay` milliseconds
- `LV_SCR_LOAD_ANIM_OVER_LEFT/RIGHT/TOP/BOTTOM` move the new screen over the current towards the given direction
- `LV_SCR_LOAD_ANIM_MOVE_LEFT/RIGHT/TOP/BOTTOM` move both the current and new screens towards the given direction
- `LV_SCR_LOAD_ANIM_FADE_ON` fade the new screen over the old screen

Setting `auto_del` to `true` will automatically delete the old screen when the animation is finished.

The new screen will become active (returned by `lv_scr_act()`) when the animation starts after `delay` time.

新屏幕也可以使用 `lv_scr_load_anim(scr, transition_type, time, delay, auto_del)` 加载动画。存在以下转换类型：

- `LV_SCR_LOAD_ANIM_NONE`: 在 `delay` 毫秒后立即切换
- `LV_SCR_LOAD_ANIM_OVER_LEFT/RIGHT/TOP/BOTTOM` 将新屏幕移动到当前的指定方向
- `LV_SCR_LOAD_ANIM_MOVE_LEFT/RIGHT/TOP/BOTTOM` 将当前屏幕和新屏幕都向给定方向移动
- `LV_SCR_LOAD_ANIM_FADE_ON` 在旧屏幕上淡出新屏幕

将 `auto_del` 设置为 `true` 将在动画完成时自动删除旧屏幕。当动画在 `delay` 时间后开始时，新屏幕将变为活动状态（由 `lv_scr_act()` 返回）。

Handling multiple displays (处理多个显示器)

Screens are created on the currently selected *default display*. The *default display* is the last registered display with `lv_disp_drv_register` or you can explicitly select a new default display using `lv_disp_set_default(disp)`.

`lv_scr_act()`, `lv_scr_load()` and `lv_scr_load_anim()` operate on the default screen.

Visit [Multi-display support](#) to learn more.

屏幕是在当前选择的默认显示上创建的。*default display* 是最后一个用 `lv_disp_drv_register` 注册的显示，或者你可以使用 `lv_disp_set_default(disp)` 明确地选择一个新的默认显示。

`lv_scr_act()`, `lv_scr_load()` 和 `lv_scr_load_anim()` 在默认屏幕上运行。

访问[多显示器支持](#)了解更多信息。

5.1.4 Parts (部分)

The widgets are built from multiple parts. For example a *Base object* uses the main and scrollbar parts but a *Slider* uses the main, the indicator and the knob parts. Parts are similar to *pseudo elements* in CSS.

The following predefined parts exist in LVGL:

- `LV_PART_MAIN` A background like rectangle*/“
- `LV_PART_SCROLLBAR` The scrollbar(s)
- `LV_PART_INDICATOR` Indicator, e.g. for slider, bar, switch, or the tick box of the checkbox

- LV_PART_KNOB Like a handle to grab to adjust the value*/
- LV_PART_SELECTED Indicate the currently selected option or section
- LV_PART_ITEMS Used if the widget has multiple similar elements (e.g. tabel cells)*/
- LV_PART_TICKS Ticks on scales e.g. for a chart or meter
- LV_PART_CURSOR Mark a specific place e.g. text area's or chart's cursor
- LV_PART_CUSTOM_FIRST Custom parts can be added from here.

The main purpose of parts to allow styling the "components" of the widgets. Therefore the parts are described in more detail in the [Style overview](#) section.

小部件由多个部分构建而成。例如，[Base object](#) 使用主部件和滚动条部件，而[Slider](#) 使用主部件、指示器和旋钮部件。部分类似于 CSS 中的伪元素。

LVGL 中存在以下预定义部分：

- LV_PART_MAIN 类似矩形的背景 */“
- LV_PART_SCROLLBAR 滚动条
- LV_PART_INDICATOR 指标，例如用于滑块、条、开关或复选框的勾选框
- LV_PART_KNOB 像手柄一样可以抓取调整值 */
- LV_PART_SELECTED 表示当前选择的选项或部分
- LV_PART_ITEMS 如果小部件有多个相似的元素（例如表格单元格） */
- LV_PART_TICKS 刻度上的刻度，例如对于图表或仪表
- LV_PART_CURSOR 标记一个特定的地方，例如文本区域或图表的光标
- LV_PART_CUSTOM_FIRST 可以从这里添加自定义部件。

部件的主要目的是允许为小部件的“组件”设置样式。因此，在[样式概述](#) 部分更详细地描述了这些部分。

5.1.5 States (状态)

The object can be in a combination of the following states:

- LV_STATE_DEFAULT Normal, released state
- LV_STATE_CHECKED Toggled or checked state
- LV_STATE_FOCUSED Focused via keypad or encoder or clicked via touchpad/mouse
- LV_STATE_FOCUS_KEY Focused via keypad or encoder but not via touchpad/mouse
- LV_STATE_EDITED Edit by an encoder
- LV_STATE_HOVERED Hovered by mouse (not supported now)
- LV_STATE_PRESSED Being pressed
- LV_STATE_SCROLLED Being scrolled
- LV_STATE_DISABLED Disabled state
- LV_STATE_USER_1 Custom state
- LV_STATE_USER_2 Custom state
- LV_STATE_USER_3 Custom state

- LV_STATE_USER_4 Custom state

The states are usually automatically changed by the library as the user presses, releases, focuses etc an object. However, the states can be changed manually too. To set or clear given state (but leave the other states untouched) use `lv_obj_add/clear_state(obj, LV_STATE_...)` In both cases ORed state values can be used as well. E.g. `lv_obj_add_state(obj, part, LV_STATE_PRESSED | LV_PRESSED_CHECKED)`.

To learn more about the states read the related section of the [Style overview](#).

对象可以处于以下状态的组合：

- LV_STATE_DEFAULT 正常，释放状态
- LV_STATE_CHECKED 切换或选中状态
- LV_STATE_FOCUSED 通过键盘或编码器聚焦或通过触摸板/鼠标点击
- LV_STATE_FOCUS_KEY 通过键盘或编码器聚焦，但不通过触摸板/鼠标聚焦
- LV_STATE_EDITED 由编码器编辑
- LV_STATE_HOVERED 鼠标悬停（现在不支持）
- LV_STATE_PRESSED 被按下
- LV_STATE_SCROLLED 正在滚动
- LV_STATE_DISABLED 禁用状态
- LV_STATE_USER_1 自定义状态
- LV_STATE_USER_2 自定义状态
- LV_STATE_USER_3 自定义状态
- LV_STATE_USER_4 自定义状态

当用户按下、释放、聚焦等对象时，库通常会自动更改状态。但是，状态也可以手动更改。要设置或清除给定状态（但保持其他状态不变），请使用 `lv_obj_add/clear_state(obj, LV_STATE_...)` 在这两种情况下，也可以使用 ORed 状态值。例如。`lv_obj_add_state(obj, part, LV_STATE_PRESSED | LV_PRESSED_CHECKED)`。

要了解有关状态的更多信息，请阅读[样式概述](#)的相关部分。

5.1.6 Snapshot (快照)

A snapshot image could be generated for object together with its children. Check details in [Snapshot](#).

可以为对象及其子对象生成快照图像。在[快照](#)中查看详细信息。

5.2 Positions, sizes, and layouts (位置、大小和布局)

5.2.1 Overview (概述)

Similarly to many other parts of LVGL, the concept of setting the coordinates was inspired by CSS. By no means a complete implementation of the standard but subsets of CSS were implemented (sometimes with minor adjustments). In shorts this means:

- the set coordinates (size, position, layouts, etc) are stored in styles
- support min-width, max-width, min-height, max-height

- have pixel, percentage, and "content" units
- x=0; y=0 coordinate means the top-left corner of the parent plus the left/top padding plus border width
- width/height means the full size, the "content area" is smaller with padding and border width
- a subset of flexbox and grid layouts are supported

与 LVGL 的许多其他部分类似，设置坐标的概念受到 CSS 的启发。绝不是标准的完整实现，而是实现了 CSS 的子集（有时会稍作调整）。简而言之，这意味着：

- 设置的坐标（大小、位置、布局等）存储在样式中
- 支持最小宽度、最大宽度、最小高度、最大高度
- 有像素、百分比和“内容”单位
- x=0; y=0 坐标表示父级的左上角加上左/上填充加上边框宽度
- 宽度/高度表示全尺寸，“内容区域”较小，填充和边框宽度
- 支持 flexbox 和网格布局的子集

Units (单位)

- pixel: Simply a position in pixels. A simple integer always means pixel. E.g. `lv_obj_set_x(btn, 10)`
- percentage: The percentage of the size of the object or its parent (depending on the property). The `lv_pct(value)` converts a value to percentage. E.g. `lv_obj_set_width(btn, lv_pct(50))`
- LV_SIZE_CONTENT: Special value to set the width/height of an object to involve all the children. It's similar to `auto` in CSS. E.g. `lv_obj_set_width(btn, LV_SIZE_CONTENT)`.
- 像素：只是一个以像素为单位的位置。一个简单的整数总是意味着像素。例如。`lv_obj_set_x(btn, 10)`
- 百分比：对象或其父对象的大小百分比（取决于属性）。`lv_pct(value)` 将值转换为百分比。例如。`lv_obj_set_width(btn, lv_pct(50))`
- LV_SIZE_CONTENT：设置对象的宽度/高度以涉及所有子项的特殊值。它类似于 CSS 中的“auto”。例如。`lv_obj_set_width(btn, LV_SIZE_CONTENT)`。

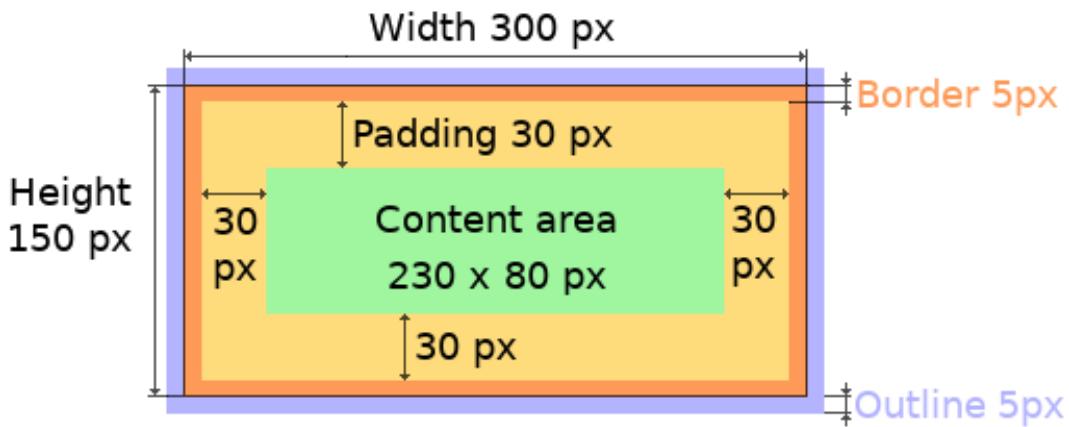
Boxing model (盒子模型)

LVGL follows CSS's `border-box` model. An object's "box" is built from the following parts:

- bounding box: the width/height of the elements.
- border width: the width of the border.
- padding: space between the sides of the object and its children.
- content: the content area which size if the bounding box reduced by the border width and the size of the paddings.

LVGL 遵循 CSS 的 `border-box` 模型。对象的“盒子”由以下部分构成：

- 边界框：元素的宽度/高度。
- 边框宽度：边框的宽度。
- 填充：对象两侧与其子对象之间的空间。
- 内容：如果边界框减少了边框宽度和填充的大小，则内容区域的大小。



LVGL 遵循 CSS 的 `border-box` 模型。对象的“盒子”由以下部分构成：

- 边界框：元素的宽度/高度。
- 边框宽度：边框的宽度。
- 填充：对象两侧与其子对象之间的空间。
- 内容：如果边界框减少了边框宽度和填充的大小，则内容区域的大小。

The border is drawn inside the bounding box. Inside the border LVGL keeps "padding size" to place the children.

The outline is drawn outside of the bounding box.

边框绘制在边界框内。在边界内 LVGL 保持“填充大小”来放置孩子。轮廓绘制在边界框之外。

Important notes (重要笔记)

This section describes special cases in which LVGL's behavior might be unexpected.

本节描述了 LVGL 的行为可能出乎意料的特殊情况。

Postponed coordinate calculation (坐标会被延迟计算)

LVGL doesn't recalculate all the coordinate changes immediately. This is done to improve performance. Instead, the objects are marked as "dirty" and before redrawing the screen LVGL checks if there are any "dirty" objects. If so it refreshes their position, size and layout.

In other words, if you need to get the any coordinate of an object and it the coordinates were just changed LVGL's needs to be forced to recalculate the coordinates. To do this call `lv_obj_update_layout(obj)`.

The size and position might depend on the parent or layout. Therefore `lv_obj_update_layout` recalculates the coordinates of all objects on the screen of `obj`.

LVGL 不会立即重新计算所有坐标变化。这样做是为了提高性能。相反，对象被标记为“脏”，并且在重绘屏幕之前 LVGL 检查是否有任何“脏”对象。如果是这样，它会刷新它们的位置、大小和布局。

换句话说，如果您需要获取对象的任何坐标并且坐标刚刚更改，则需要强制 LVGL 重新计算坐标。为此调用 `lv_obj_update_layout(obj)`。

大小和位置可能取决于父级或布局。因此 `lv_obj_update_layout` 重新计算 `obj` 屏幕上所有对象的坐标。

Removing styles (删除样式)

As it's described in the [Using styles](#) section the coordinates can be set via style properties too. To be more precise under the hood every style coordinate related property is stored as style a property. If you use `lv_obj_set_x(obj, 20)` LVGL saves $x=20$ in the local style of the object.

It's an internal mechanism and doesn't matter much as you use LVGL. However, there is one case in which you need to aware of that. If the style(s) of an object are removed by

正如[使用样式](#)部分所述，坐标也可以通过样式属性设置。更准确地说，每个与样式坐标相关的属性都存储为样式属性。如果你使用 `lv_obj_set_x(obj, 20)` LVGL 将 $x=20$ 保存在对象的本地样式中。

这是一种内部机制，与您使用 LVGL 无关。但是，在一种情况下，您需要了解这一点。如果对象的样式被删除

```
lv_obj_remove_style_all(obj)
```

or

或者

```
lv_obj_remove_style(obj, NULL, LV_PART_MAIN);
```

the earlier set coordinates will be removed as well.

For example:

先前设置的坐标也将被删除。

例如：

```
/*The size of obj1 will be set back to the default in the end*/
lv_obj_set_size(obj1, 200, 100); /*Now obj1 has 200;100 size*/
lv_obj_remove_style_all(obj1); /*It removes the set sizes*/

/*obj2 will have 200;100 size in the end */
lv_obj_remove_style_all(obj2);
lv_obj_set_size(obj2, 200, 100);
```

5.2.2 Position (位置)

Simple way (最简单的方法)

To simple set the x and y coordinates of an object use

要简单设置对象的 x 和 y 坐标，请使用

```
lv_obj_set_x(obj, 10);
lv_obj_set_y(obj, 20);
lv_obj_set_pos(obj, 10, 20);           //Or in one function
```

By default the the x and y coordinates are measured from the top left corner of the parent's content area. For example if the parent has 5 pixels padding on every side, the above code will place `obj` at (15, 25) because the content area starts after the padding.

If percentage values are calculated from the parents content area size.

默认情况下，x 和 y 坐标是从父内容区域的左上角开始测量的。例如，如果父级每边有 5 个像素的填充，上面的代码会将 obj 放置在 (15, 25) 处，因为内容区域在填充之后开始。

如果百分比值是根据父内容区域大小计算的。

```
lv_obj_set_x(btn, lv_pct(10)); //x = 10 % of parent content area width
```

Align (对齐)

In some cases it's convenient to change the origin of the positioning from the the default top left. If the origin is changed e.g. to bottom-right, the (0,0) position means: align to the bottom-right corner. To change the origin use:

在某些情况下，从默认的左上角更改定位的原点会很方便。如果原点改变，例如到右下角，(0,0) 位置表示：与右下角对齐。要更改原点使用：

```
lv_obj_set_align(obj, align);
```

To change the alignment and set new coordinates:

要更改对齐方式并设置新坐标：

```
lv_obj_align(obj, align, x, y);
```

The following alignment options can be used:

- 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

It quite common to align a children to the center of its parent, there fore is a dedicated function for it:

可以使用以下对齐选项：

- 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

将孩子对齐到其父母的中心是很常见的，因此有一个专门的功能：

```
lv_obj_center(obj);
//Has the same effect
lv_obj_align(obj, LV_ALIGN_CENTER, 0, 0);
```

If the parent's size changes the set alignment and position of the children is applied again automatically.

The functions introduced above aligns the object to its parent. However it's also possible to align an object to an arbitrary object.

如果父项的大小更改，则会自动再次应用子项的设置对齐方式和位置。

上面介绍的函数将对象与其父对象对齐。但是，也可以将对象与任意对象对齐。

```
lv_obj_align_to(obj_to_align, reference_obj, align, x, y);
```

Besides the alignments options above the following can be used to align the object outside of the reference object:

- 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

For example to align a label above a button and center the label horizontally:

除了上面的对齐选项之外，以下选项还可用于对齐参考对象之外的对象：

- 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

例如，在按钮上方对齐标签并使标签水平居中：

```
lv_obj_align_to(label, btn, LV_ALIGN_OUT_TOP_MID, 0, -10);
```

Note that - unlike with `lv_obj_align()` - `lv_obj_align_to()` can not realign the object if its coordinates or the reference object's coordinates changes.

请注意 - 与 `lv_obj_align()` 不同 - `lv_obj_align_to()` 无法重新对齐对象，如果其坐标或参考对象的坐标发生变化。

5.2.3 Size (大小)

Simple way (最简单的方法)

The width and the height of an object can be set easily as well:

对象的宽度和高度也可以轻松设置：

```
lv_obj_set_width(obj, 200);
lv_obj_set_height(obj, 100);
lv_obj_set_size(obj, 200, 100);           //Or in one function
```

Percentage values are calculated based on the parent's content area size. For example to set the object's height to the screen height:

百分比值是根据父内容区域的大小计算的。例如将对象的高度设置为屏幕高度：

```
lv_obj_set_height(obj, lv_pct(100));
```

Size setting supports a value: `LV_SIZE_CONTENT`. It means the object's size in the respective direction will be set to the size of its children. Note that only children on the right and bottom will be considered and children on the top and left remain cropped. This limitation makes the behavior more predictable.

Objects with `LV_OBJ_FLAG_HIDDEN` or `LV_OBJ_FLAG_FLOATING` will be ignored by the `LV_SIZE_CONTENT` calculation.

The above functions set the size of the bounding box of the object but the size of the content area can be set as well. It means the object's bounding box will be larger with the paddings than the set size.

大小设置支持一个值：`LV_SIZE_CONTENT`。这意味着对象在相应方向上的大小将设置为其子对象的大小。
请注意，只会考虑右侧和底部的子项，而顶部和左侧的子项仍会被裁剪。此限制使行为更可预测。

带有 `LV_OBJ_FLAG_HIDDEN` 或 `LV_OBJ_FLAG_FLOATING` 的对象将被 `LV_SIZE_CONTENT` 计算忽略。

上述函数设置对象边界框的大小，但也可以设置内容区域的大小。这意味着对象的边界框将比设置的大小更大。

```
lv_obj_set_content_width(obj, 50); //The actual width: padding left + 50 + padding ↵right
lv_obj_set_content_height(obj, 30); //The actual width: padding top + 30 + padding ↵bottom
```

The size of the bounding box and the content area can be get with the following functions:

可以使用以下函数获取边界框和内容区域的大小：

```
lv_coord_t w = lv_obj_get_width(obj);
lv_coord_t h = lv_obj_get_height(obj);
lv_coord_t content_w = lv_obj_get_content_width(obj);
lv_coord_t content_h = lv_obj_get_content_height(obj);
```

5.2.4 Using styles (使用样式)

Under the hood the position, size and alignment properties are style properties. The above described "simple functions" hide the style related code for the sake of simplicity and set the position, size, and alignment properties in the local styles of the object.

However, using styles as to set the coordinates has some great advantages:

- It makes it easy to set the width/height/etc for several objects together. E.g. make all the sliders 100x10 pixels sized.
- It also makes possible to modify the values in one place.
- The values can be overwritten by other styles. For example `style_btn` makes the object `100x50` by default but adding `style_full_width` overwrites only the width of the object.
- The object can have different position or size in different state. E.g. 100 px wide in `LV_STATE_DEFAULT` but 120 px in `LV_STATE_PRESSED`.
- Style transitions can be used to make the coordinate changes smooth.

在驱动中，位置、大小和对齐属性是样式属性。上面描述的“简单函数”为了简单起见隐藏了样式相关的代码，并在对象的局部样式中设置了位置、大小和对齐属性。

但是，使用样式设置坐标有一些很大的优点：

- 可以轻松地为多个对象设置宽度/高度/等。例如。使所有滑块的大小为 100x10 像素。
- 还可以在一处修改值。
- 这些值可以被其他样式覆盖。例如 `style_btn` 默认使对象 `100x50`，但添加 `style_full_width` 只会覆盖对象的宽度。
- 物体在不同状态下可以有不同的位置或大小。例如。`LV_STATE_DEFAULT` 中的宽度为 100 像素，而 `LV_STATE_PRESSED` 中的宽度为 120 像素。
- 样式转换可用于使坐标变化平滑。

Here are some examples to set an object's size using a style:

以下是一些使用样式设置对象大小的示例：

```
static lv_style_t style;
lv_style_init(&style);
lv_style_set_width(&style, 100);

lv_obj_t * btn = lv_btn_create(lv_scr_act());
lv_obj_add_style(btn, &style, LV_PART_MAIN);
```

As you will see below there are some other great features of size and position setting. However, to keep the LVGL's API lean only the most common coordinate setting features have a "simple" version and the more complex features can be used via styles.

正如您将在下面看到的，还有一些其他重要的尺寸和位置设置功能。然而，为了保持 LVGL 的 API 精简，只有最常见的坐标设置功能有一个“简单”版本，更复杂的功能可以通过样式使用。

5.2.5 Translation (风格样式转换)

Let's say there are 3 buttons next to each other. Their position is set as described above. Now you want to move a button up a little when it's pressed.

One way to achieve this is setting a new Y coordinate for pressed state:

假设有 3 个按钮彼此相邻。它们的位置如上所述设置。现在您想在按下按钮时将其向上移动一点。

实现此目的的一种方法是为按下状态设置新的 Y 坐标：

```
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);
```

It works but it's not really flexible because the pressed coordinate is hard-coded. If the buttons are not at y=100 **style_pressed** won't work as expected. To solve this translations can be used:

它可以工作，但不是很灵活，因为按下的坐标是硬编码的。如果按钮不在 y=100 处，**style_pressed** 将不会按预期工作。要解决这个问题，可以参考使用以下转换：

```
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);
```

Translation is applied from the current position of the object.

Percentage values can be used in translations as well. The percentage is relative to the size of the object (and not to the size of the parent). For example `lv_pct(50)` will move the object with half of its width/height.

The translation is applied after the layouts are calculated. Therefore, even the layouted objects' position can be translated.

The translation actually moves the object. It means it makes the scrollbars and `LV_SIZE_CONTENT` sized objects react to the position change.

从对象的当前位置开始应用平移。

百分比值也可用于翻译。百分比是相对于对象的大小（而不是父对象的大小）。例如，`lv_pct(50)` 将移动对象的宽度/高度的一半。

在计算布局后应用翻译。因此，甚至可以平移布局对象的位置。

平移实际上移动了对象。这意味着它使滚动条和 `LV_SIZE_CONTENT` 大小的对象对位置变化做出反应。

5.2.6 Transformation (转换)

Similarly to the position the size can be changed relative to the current size as well. The transformed width and height are added on both sides of the object. This means 10 px transformed width makes the object 2x10 pixel wider.

Unlike position translation, the size transformation doesn't make the object "really" larger. In other words scrollbars, layouts, `LV_SIZE_CONTENT` will not consider the transformed size. Hence size transformation if "only" a visual effect.

This code makes the a button larger when it's pressed:

与位置类似，大小也可以相对于当前大小进行更改。转换后的宽度和高度会添加到对象的两侧。这意味着 10 px 转换宽度使对象更宽 2x10 像素。

与位置平移不同，尺寸变换不会使对象“真正”变大。换句话说，滚动条、布局、`LV_SIZE_CONTENT` 不会考虑转换后的大小。因此，如果“仅”是视觉效果，则尺寸转换。

此代码使下面示例的按钮在按下时变大：

```
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);
```

Min and Max size (最小和最大尺寸)

Similarly to CSS, LVGL also support `min-width`, `max-width`, `min-height` and `max-height`. These are limits preventing an object's size to be smaller/larger than these values. They are especially useful if the size is set by percentage or `LV_SIZE_CONTENT`.

与 CSS 类似，LVGL 也支持 `min-width`、`max-width`、`min-height` 和 `max-height`。这些是防止对象的大小小于/大于这些值的限制。如果大小按百分比或 `LV_SIZE_CONTENT` 设置，则它们特别有用。

```
static lv_style_t style_max_height;
lv_style_init(&style_max_height);
lv_style_set_y(&style_max_height, 200);

lv_obj_set_height(obj, lv_pct(100));
lv_obj_add_style(obj, &style_max_height, LV_STATE_DEFAULT); //Limit the height to
→200 px
```

Percentage values can be used as well which are relative to the size of the parent's content area size.

也可以使用与父内容区域大小相关的百分比值。

```
static lv_style_t style_max_height;
lv_style_init(&style_max_height);
lv_style_set_y(&style_max_height, lv_pct(50));
```

(下页继续)

(续上页)

```
lv_obj_set_height(obj, lv_pct(100));
lv_obj_add_style(obj, &style_max_height, LV_STATE_DEFAULT); //Limit the height to
//half parent height
```

5.2.7 Layout (布局)

Overview (概述)

Layouts can update the position and size of an object's children. They can be used to automatically arrange the children into a line or column, or in much more complicated forms.

The position and size set by the layout overwrites the "normal" x, y, width, and height settings.

There is only one function that is the same for every layout: `lv_obj_set_layout(obj, <LAYOUT_NAME>)` sets the layout on an object. For the further settings of the parent and children see the documentations of the given layout.

布局可以更新对象子项的位置和大小。它们可用于自动将子项排列成一行或一列，或者以更复杂的形式排列。

布局设置的位置和大小会覆盖“正常”的x、y、宽度和高度设置。

每个布局只有一个相同的函数: `lv_obj_set_layout(obj, <LAYOUT_NAME>)` 在对象上设置布局。有关父级和子级的进一步设置，请参阅给定布局的文档。

Built-in layout

LVGL comes with two very powerful layouts:

- Flexbox
- Grid

Both are heavily inspired by the CSS layouts with the same name.

LVGL 带有两个非常强大的布局:

- Flexbox(弹性伸缩盒)
- Grid(网格)

两者都深受 CSS 布局的启发。

Flags (标志)

There are some flags that can be used on object to affect how they behave with layouts:

- `LV_OBJ_FLAG_HIDDEN` Hidden object are ignored from layout calculations.
- `LV_OBJ_FLAG_IGNORE_LAYOUT` The object is simply ignored by the layouts. Its coordinates can be set as usual.
- `LV_OBJ_FLAG_FLOATING` Same as `LV_OBJ_FLAG_IGNORE_LAYOUT` but the object with `LV_OBJ_FLAG_FLOATING` will be ignored from `LV_SIZE_CONTENT` calculations.

These flags can be added/removed with `lv_obj_add/clear_flag(obj, FLAG)`;

有一些标志可用于对象以影响它们在布局中的行为:

- `LV_OBJ_FLAG_HIDDEN` 隐藏对象从布局计算中被忽略。
- `LV_OBJ_FLAG_IGNORE_LAYOUT` 该对象被布局简单地忽略。它的坐标可以照常设置。
- `LV_OBJ_FLAG_FLOATING` 与 `LV_OBJ_FLAG_IGNORE_LAYOUT` 相同，但带有 `LV_OBJ_FLAG_FLOATING` 的对象将在 `LV_SIZE_CONTENT` 计算中被忽略。

可以使用 `lv_obj_add/clear_flag(obj, FLAG)`; 添加/删除这些标志

Adding new layouts (添加新布局)

LVGL can be freely extended by a custom layouts like this:

LVGL 可以通过这样的自定义布局自由扩展：

```
uint32_t MY_LAYOUT;

...
MY_LAYOUT = lv_layout_register(my_layout_update, &user_data);
...

void my_layout_update(lv_obj_t * obj, void * user_data)
{
    /*Will be called automatically if required to reposition/resize the children
    ↵of "obj" */
}
```

Custom style properties can be added too that can be get and used in the update callback. For example:

也可以添加可以在更新回调中获取和使用的自定义样式属性。例如：

```
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);
}
```

5.2.8 Examples

5.3 Styles (风格样式)

Styles are used to set the appearance of the objects. Styles in lvgl are heavily inspired by CSS. The concept in nutshell is as follows:

- A style is an `lv_style_t` variable which can hold properties like border width, text color and so on. It's similar to a `class` in CSS.

- Styles can be assigned to objects to change their appearance. During the assignment the target part (*pseudo element* in CSS) and target state (*pseudo class*) can be specified. For example one can add `style_blue` to the knob of a slider when it's in pressed state.
- The same style can be used by any number of objects.
- Styles can be cascaded which means multiple styles can be assigned to an object and each style can have different properties. Therefore not all properties have to be specified in style. LVGL will look for a property until a style defines it or use a default if it's not specified by any of the styles. For example `style_btn` can result in a default gray button and `style_btn_red` can add only a `background-color=red` to overwrite the background color.
- Later added styles have higher precedence. It means if a property is specified in two styles the later added will be used.
- Some properties (e.g. text color) can be inherited from the parent(s) if it's not specified in the object.
- Objects can have local styles that have higher precedence than "normal" styles.
- Unlike CSS (where pseudo-classes describe different states, e.g. `:focus`), in LVGL a property is assigned to a given state.
- Transitions can be applied when the object changes state.

Styles 用于设置对象的外观。lvgl 中的样式很大程度上受到 CSS 的启发。简而言之，其概念如下：

- 样式是一个 `lv_style_t` 变量，它可以保存边框宽度、文本颜色等属性。它类似于 CSS 中的“类”。
- 可以将样式分配给对象以更改其外观。在赋值过程中，可以指定目标部分 (CSS 中的 *pseudo element*) 和目标状态 (*pseudo class*)。

例如，当滑块处于按下状态时，可以将 “`style_blue`” 添加到滑块的旋钮。

- 任何数量的对象都可以使用相同的样式。
- 样式可以级联，这意味着可以将多个样式分配给一个对象，并且每个样式可以具有不同的属性。

因此，并非所有属性都必须在样式中指定。LVGL 将寻找一个属性，直到一个样式定义它，或者如果它没有被任何样式指定，则使用默认值。

例如，`style_btn` 可以导致默认的灰色按钮，而 `style_btn_red` 只能添加一个 `background-color=red` 来覆盖背景颜色。

- 后来添加的样式具有更高的优先级。这意味着如果在两种样式中指定了一个属性，则将使用稍后添加的样式。
- 如果对象中未指定某些属性（例如文本颜色），则可以从父级继承。
- 对象可以具有比“正常”样式具有更高优先级的本地样式。
- 与 CSS（伪类描述不同的状态，例如：`:focus`）不同，在 LVGL 中，属性被分配给给定的状态。
- 当对象改变状态时可以应用转换。

5.3.1 States (状态)

The objects can be in the combination of the following states:

- LV_STATE_DEFAULT (0x0000) Normal, released state
- LV_STATE_CHECKED (0x0001) Toggled or checked state
- LV_STATE_FOCUSED (0x0002) Focused via keypad or encoder or clicked via touchpad/mouse
- LV_STATE_FOCUS_KEY (0x0004) Focused via keypad or encoder but not via touchpad/mouse
- LV_STATE_EDITED (0x0008) Edit by an encoder
- LV_STATE_HOVERED (0x0010) Hovered by mouse (not supported now)
- LV_STATE_PRESSED (0x0020) Being pressed
- LV_STATE_SCROLLLED (0x0040) Being scrolled
- LV_STATE_DISABLED (0x0080) Disabled state
- LV_STATE_USER_1 (0x1000) Custom state
- LV_STATE_USER_2 (0x2000) Custom state
- LV_STATE_USER_3 (0x4000) Custom state
- LV_STATE_USER_4 (0x8000) Custom state

The combination states the object can be focused and pressed at the same time. This is represented as `LV_STATE_FOCUSED | LV_STATE_PRESSED`.

The style can be added to any state and state combination. For example, setting a different background color for default and pressed state. 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)

对象可以处于以下状态的组合:

- LV_STATE_DEFAULT (0x0000) 正常, 释放状态
- LV_STATE_CHECKED (0x0001) 切换或检查状态
- LV_STATE_FOCUSED (0x0002) 通过键盘或编码器聚焦或通过触摸板/鼠标点击
- LV_STATE_FOCUS_KEY (0x0004) 通过键盘或编码器聚焦, 但不通过触摸板/鼠标聚焦
- LV_STATE_EDITED (0x0008) 由编码器编辑
- LV_STATE_HOVERED (0x0010) 鼠标悬停 (现在不支持)
- LV_STATE_PRESSED (0x0020) 被按下
- LV_STATE_SCROLLLED (0x0040) 正在滚动
- LV_STATE_DISABLED (0x0080) 禁用状态
- LV_STATE_USER_1 (0x1000) 自定义状态
- LV_STATE_USER_2 (0x2000) 自定义状态
- LV_STATE_USER_3 (0x4000) 自定义状态
- LV_STATE_USER_4 (0x8000) 自定义状态

该组合表示可以同时聚焦和按下对象。这表示为 `LV_STATE_FOCUSED | LV_STATE_PRESSED`。

样式可以添加到任何状态和状态组合。例如，为默认和按下状态设置不同的背景颜色。如果属性未在状态中定义，则将使用最佳匹配状态的属性。通常这意味着使用带有“`LV_STATE_DEFAULT`”的属性。如果即使为默认状态也未设置该属性，则将使用默认值。（见后）

But what does the "best matching state's property" really mean? States have a precedence which is shown by their value (see in the above list). A higher value means higher precedence. To determine which state's property to use let's take an example. Imagine the background color is defined like this:

- `LV_STATE_DEFAULT`: white
- `LV_STATE_PRESSED`: gray
- `LV_STATE_FOCUSED`: red

1. By the default the object is in default state, so it's a simple case: the property is perfectly defined in the object's current state as white.
2. When the object is pressed there are 2 related properties: default with white (default is related to every state) and pressed with gray. The pressed state has `0x0020` precedence which is higher than the default state's `0x0000` precedence, so gray color will be used.
3. When the object is focused the same thing happens as in pressed state and red color will be used. (Focused state has higher precedence than default state).
4. When the object is focused and pressed both gray and red would work, but the pressed state has higher precedence than focused so gray color will be used.
5. It's possible to set e.g rose color for `LV_STATE_PRESSED | LV_STATE_FOCUSED`. In this case, this combined state has $0x0020 + 0x0002 = 0x0022$ precedence, which is higher than the pressed state's precedence so rose color would be used.
6. When the object is in checked state there is no property to set the background color for this state. So for lack of a better option, the object remains white from the default state's property.

但是“最匹配的国家财产”到底是什么意思呢？状态具有优先级，由它们的值显示（参见上面的列表）。更高的值意味着更高的优先级。为了确定使用哪个状态的属性，让我们举个例子。想象一下，背景颜色是这样定义的：

- `LV_STATE_DEFAULT`: 白色
- `LV_STATE_PRESSED`: 灰色
- `LV_STATE_FOCUSED`: 红色

1. 默认情况下，对象处于默认状态，所以这是一个简单的情况：属性在对象当前状态下完美定义为白色。
2. 当对象被按下时有 2 个相关属性：默认为白色（默认与每个状态相关）和按下为灰色。按下状态的优先级为 `0x0020`，高于默认状态的 `0x0000` 优先级，因此将使用灰色。
3. 当物体聚焦时，发生与按下状态相同的事情，将使用红色。（焦点状态比默认状态具有更高的优先级）。
4. 当物体聚焦并按下时，灰色和红色都可以工作，但按下状态的优先级高于聚焦状态，因此将使用灰色。
5. 可以为 `LV_STATE_PRESSED | LV_STATE_FOCUSED` 设置例如玫瑰色。在这种情况下，此组合状态的优先级为 $0x0020 + 0x0002 = 0x0022$ ，高于按下状态的优先级，因此将使用玫瑰色。
6. 当对象处于选中状态时，没有设置此状态的背景颜色的属性。因此，由于缺乏更好的选择，对象从默认状态的属性中保持白色。

Some practical notes:

- The precedence (value) of states is quite intuitive and it's something the user would expect naturally. E.g. if an object is focused the user will still want to see if it's pressed, therefore pressed state has a higher precedence. If the focused state had a higher precedence it would overwrite the pressed color.
- If you want to set a property for all states (e.g. red background color) just set it for the default state. If the object can't find a property for its current state it will fall back to the default state's property.
- Use ORed states to describe the properties for complex cases. (E.g. pressed + checked + focused)
- It might be a good idea to use different style elements for different states. For example, finding background colors for released, pressed, checked + pressed, focused, focused + pressed, focused + pressed + checked, etc states is quite difficult. Instead, for example, use the background color for pressed and checked states and indicate the focused state with a different border color.

一些实用的注意事项：

- 状态的优先级（值）非常直观，这是用户自然期望的。例如。如果一个对象被聚焦，用户仍然希望查看它是否被按下，因此按下状态具有更高的优先级。如果聚焦状态具有更高的优先级，它将覆盖按下的颜色。
- 如果您想为所有状态设置一个属性（例如红色背景色），只需将其设置为默认状态即可。如果对象找不到当前状态的属性，它将回退到默认状态的属性。
- 使用 ORed 状态来描述复杂情况的属性。（例如按下 + 选中 + 聚焦）
- 为不同的状态使用不同的样式元素可能是个好主意。例如，为 released、pressed、checked+pressed、focused、focused+pressed、focused+pressed+checked 等状态寻找背景颜色是相当困难的。相反，例如，对按下和选中状态使用背景颜色，并使用不同的边框颜色指示聚焦状态。

5.3.2 Cascading styles (层叠样式)

It's not required to set all the properties in one style. It's possible to add more styles to an object and let the later added style to modify or extend appearance. For example, create a general gray button style and create a new for red buttons where only the new background color is set.

This is much like in CSS when used classes are listed like `<div class=".btn .btn-red">`.

Styles added later have precedence over ones set earlier. So in the gray/red button example above, the normal button style should be added first and the red style second. However, the precedence coming from states are still taken into account. So let's examine the following case:

- the basic button style defines dark-gray color for default state and light-gray color pressed state
- the red button style defines the background color as red only in the default state

In this case, when the button is released (it's in default state) it will be red because a perfect match is found in the most recently added style (red). When the button is pressed the light-gray color is a better match because it describes the current state perfectly, so the button will be light-gray.

不需要在一种样式中设置所有属性。可以向对象添加更多样式，并让稍后添加的样式修改或扩展外观。例如，创建一个通用的灰色按钮样式并为红色按钮创建一个新的，其中只设置了新的背景颜色。

这很像在 CSS 中，当使用的类像 `<div class=".btn .btn-red">` 一样列出时。

后来添加的样式优先于之前设置的样式。所以在上面的灰色/红色按钮示例中，应该首先添加普通按钮样式，然后添加红色样式。但是，仍然考虑来自州的优先权。因此，让我们检查以下案例：

- 基本按钮样式定义了默认状态的深灰色和浅灰色按下状态
- 红色按钮样式仅在默认状态下将背景颜色定义为红色

在这种情况下，当按钮被释放（处于默认状态）时，它将是红色的，因为在最近添加的样式（红色）中找到了完美匹配。当按下按钮时，浅灰色更适合，因为它完美地描述了当前状态，所以按钮将是浅灰色。

5.3.3 Inheritance (继承)

Some properties (typically that are related to texts) can be inherited from the parent object's styles. Inheritance is applied only if the given property is not set in the object's styles (even in default state). In this case, if the property is inheritable, the property's value will be searched in the parents too until an object specifies a value for the property. The parents will use their own state to determine the value. So if a button is pressed, and the text color comes from here, the pressed text color will be used.

某些属性（通常与文本相关）可以从父对象的样式继承。仅当未在对象的样式中设置给定属性时（即使在默认状态下），才应用继承。在这种情况下，如果该属性是可继承的，则该属性的值也将在父项中搜索，直到一个对象为该属性指定了一个值。父母将使用自己的状态来确定该值。因此，如果按下按钮，并且文本颜色来自此处，则将使用按下的文本颜色。

5.3.4 Parts

Objects can have *parts* which can have their own styles.

The following predefined parts exist in LVGL:

- LV_PART_MAIN A background like rectangle*/
- LV_PART_SCROLLBAR The scrollbar(s)
- LV_PART_INDICATOR Indicator, e.g. for slider, bar, switch, or the tick box of the checkbox
- LV_PART_KNOB Like a handle to grab to adjust the value*/
- LV_PART_SELECTED Indicate the currently selected option or section
- LV_PART_ITEMS Used if the widget has multiple similar elements (e.g. table cells)*/
- LV_PART_TICKS Ticks on scales e.g. for a chart or meter
- LV_PART_CURSOR Mark a specific place e.g. text area's or chart's cursor
- LV_PART_CUSTOM_FIRST Custom parts can be added from here.

For example a *Slider* has three parts:

- Background
- Indicator
- Knob

It means the all three parts of the slider can have their own styles. See later how to add style styles to objects and parts.

对象可以有部分，它们可以有自己的样式。

LVGL 中存在以下预定义部分：

- LV_PART_MAIN 类似矩形的背景 */
- LV_PART_SCROLLBAR 滚动条
- LV_PART_INDICATOR 指标，例如用于滑块、条、开关或复选框的勾选框
- LV_PART_KNOB 像手柄一样可以抓取调整值 */
- LV_PART_SELECTED 表示当前选择的选项或部分

- LV_PART_ITEMS 如果小部件具有多个相似元素（例如表格单元格）*/
- LV_PART_TICKS 刻度上的刻度，例如对于图表或仪表
- LV_PART_CURSOR 标记一个特定的地方，例如文本区域或图表的光标
- LV_PART_CUSTOM_FIRST 可以从这里添加自定义部件。

例如一个 *Slider* 包含三个部分：

- 背景
- 印度人
- 旋钮

这意味着滑块的所有三个部分都可以有自己的样式。稍后请参阅如何向对象和部件添加样式样式。

5.3.5 Initialize styles and set/get properties (初始化样式和设置/获取属性)

Styles are stored in `lv_style_t` variables. Style variables should be `static`, global or dynamically allocated. In other words they can not be local variables in functions which are destroyed when the function exists. Before using a style it should be initialized with `lv_style_init(&my_style)`. After initializing the style properties can be set or added to it.

Property set functions looks like this: `lv_style_set_<property_name>(&style, <value>)`; For example:

样式存储在 `lv_style_t` 变量中。样式变量应该是“静态”、全局或动态分配的。换句话说，它们不能是函数中的局部变量，当函数存在时它们会被销毁。在使用样式之前，它应该用 `lv_style_init(&my_style)` 进行初始化。初始化后，可以设置或添加样式属性。

属性集函数看起来像这样: `lv_style_set_<property_name>(&style, <value>)`; 例如:

```
static lv_style_t style_btn;
lv_style_init(&style_btn);
lv_style_set_bg_color(&style_btn, lv_color_grey());
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_color_red());
lv_style_set_bg_opa(&style_btn_red, LV_OPA_COVER);
```

To remove a property use:

要删除属性，请使用：

```
lv_style_remove_prop(&style, LV_STYLE_BG_COLOR);
```

To get a property's value from a style:

从样式中获取属性的值：

```
lv_style_value_t v;
lv_res_t res = lv_style_get_prop(&style, LV_STYLE_BG_COLOR, &v);
if(res == LV_RES_OK) { /*Found*/
    do_something(v.color);
}
```

`lv_style_value_t` has 3 fields:

- `num` for integer, boolean and opacity properties
- `color` for color properties
- `ptr` for pointer properties

To reset a style (free all its data) use

`lv_style_value_t` 有 3 个字段:

- `num` 用于整数、布尔值和不透明度属性
- 颜色属性的 `color`
- 指针属性的 `ptr`

要重置样式 (释放其所有数据), 请使用

```
lv_style_reset(&style);
```

5.3.6 Add and remove styles to a widget (向部件添加和删除样式)

A style on its own is not that useful, it needs to be assigned to an object to take effect.

一个样式本身并没有那么有用, 它需要分配给一个对象才能生效。

Add styles (添加样式)

To add a style to an object use `lv_obj_add_style(obj, &style, <selector>)`. `<selector>` is an OR-ed value of parts and state to which the style should be added. Some examples:

- `LV_PART_MAIN | LV_STATE_DEFAULT`
- `LV_STATE_PRESSED`: The main part in pressed state. `LV_PART_MAIN` can be omitted
- `LV_PART_SCROLLBAR`: The scrollbar part in the default state. `LV_STATE_DEFAULT` can be omitted.
- `LV_PART_SCROLLBAR | LV_STATE_SCROLLED`: The scrollbar part when the object is being scrolled
- `0` Same as `LV_PART_MAIN | LV_STATE_DEFAULT`.
- `LV_PART_INDICATOR | LV_STATE_PRESSED | LV_STATE_CHECKED` The indicator part when the object is pressed and checked at the same time.

Using `lv_obj_add_style`:

要向对象添加样式, 请使用 `lv_obj_add_style(obj, &style, <selector>)`。`<selector>` 是应添加样式的部分和状态的 OR 值。一些例子:

- `LV_PART_MAIN | LV_STATE_DEFAULT`
- `LV_STATE_PRESSED`: 处于按下状态的主要部分。`LV_PART_MAIN` 可以省略
- `LV_PART_SCROLLBAR`: 默认状态下的滚动条部分。`LV_STATE_DEFAULT` 可以省略。
- `LV_PART_SCROLLBAR | LV_STATE_SCROLLED`: 对象滚动时的滚动条部分
- `0` 与 `LV_PART_MAIN | LV_STATE_DEFAULT` 相同。
- `LV_PART_INDICATOR | LV_STATE_PRESSED | LV_STATE_CHECKED` 同时按下和检查对象时的指示器部分。

使用 `lv_obj_add_style` 的示例:

```
lv_obj_add_style(btn, &style_btn, 0);                                /
↳ *Default button style*
lv_obj_add_style(btn, &btn_red, LV_STATE_PRESSED); /*Overwrite only a some colors to red when pressed*/
```

Remove styles (删除样式)

To remove all styles from an object use `lv_obj_remove_style_all(obj)`.

To remove specific styles use `lv_obj_remove_style(obj, style, selector)`. This function will remove `style` only if the `selector` matches with the `selector` used in `lv_obj_add_style`. `style` can be `NULL` to check only the `selector` and remove all matching styles. The `selector` can use the `LV_STATE_ANY` and `LV_PART_ANY` values to remove the style with any state or part.

要从对象中删除所有样式，请使用 `lv_obj_remove_style_all(obj)`。

要删除特定样式，请使用 `lv_obj_remove_style(obj, style, selector)`。仅当 `selector` 与 `lv_obj_add_style` 中使用的 `selector` 匹配时，此函数才会删除 `style`。`style` 可以是 `NULL` 以仅检查 `selector` 并删除所有匹配的样式。`selector` 可以使用 `LV_STATE_ANY` 和 `LV_PART_ANY` 值来删除具有任何状态或部分的样式。

Report style changes (通知样式更改)

If a style which is already assigned to object changes (i.e. a property is added or changed) the objects using that style should be notified. There are 3 options to do this:

1. If you know that the changed properties can be applied by a simple redraw (e.g. color or opacity changes) just call `lv_obj_invalidate(obj)` or `lv_obj_invalidate(lv_scr_act())`.
2. If more complex style properties were changed or added, and you know which object(s) are affected by that style call `lv_obj_refresh_style(obj, part, property)`. To refresh all parts and properties use `lv_obj_refresh_style(obj, LV_PART_ANY, LV_STYLE_PROP_ANY)`.
3. To make LVGL check all objects to see whether they use the style and refresh them when needed call `lv_obj_report_style_change(&style)`. If `style` is `NULL` all objects will be notified about the style change.

如果已分配给对象的样式发生更改（即添加或更改属性），则应通知使用该样式的对象。有 3 个选项可以执行此操作：

1. 如果您知道更改的属性可以通过简单的重绘（例如颜色或不透明度更改）应用，只需调用 `lv_obj_invalidate(obj)` 或 `lv_obj_invalidate(lv_scr_act())`。
2. 如果更改或添加了更复杂的样式属性，并且您知道哪些对象受该样式影响，则调用 `lv_obj_refresh_style(obj, part, property)`。要刷新所有部件和属性，请使用 `lv_obj_refresh_style(obj, LV_PART_ANY, LV_STYLE_PROP_ANY)`。
3. 要让 LVGL 检查所有对象是否使用该样式并在需要时刷新它们，请调用 `lv_obj_report_style_change(&style)`。如果 `style` 为 `NULL`，所有对象都会收到有关样式更改的通知。

Get a property's value on an object (获取对象的属性值)

To get a final value of property - considering cascading, inheritance, local styles and transitions (see below) - get functions like this can be used: `lv_obj_get_style_<property_name>(obj, <part>)`. These functions uses the object's current state and if no better candidate returns a default value. For example:

要获得属性的最终值——考虑级联、继承、局部样式和转换（见下文）——可以使用这样的获取函数：`lv_obj_get_style_<property_name>(obj, <part>)`。这些函数使用对象的当前状态，如果没有更好的候选者，则返回默认值。例如：

```
lv_color_t color = lv_obj_get_style_bg_color(btn, LV_PART_MAIN);
```

5.3.7 Local styles (本地样式)

Besides "normal" styles, the objects can store local styles too. This concept is similar to inline styles in CSS (e.g. `<div style="color:red">`) with some modification.

So local styles are like normal styles but they can't be shared among other objects. If used, local styles are allocated automatically, and freed when the object is deleted. They are useful to add local customization to the object.

Unlike in CSS, in LVGL local styles can be assigned to states (*pseudo-classes*) and parts (*pseudo-elements*).

To set a local property use functions like `lv_obj_set_style_local_<property_name>(obj, <value>, <selector>)`; For example:

除了“正常”样式，对象也可以存储本地样式。这个概念类似于 CSS 中的内联样式（例如`<div style="color:red">`），但做了一些修改。

所以本地样式就像普通样式，但它们不能在其他对象之间共享。如果使用，本地样式会自动分配，并在删除对象时释放。它们对于向对象添加本地自定义很有用。

与 CSS 不同，在 LVGL 中，可以将局部样式分配给状态 (*pseudo-classes*) 和部分 (*pseudo-elements*)。

要设置本地属性，请使用诸如 `lv_obj_set_style_local_<property_name>(obj, ,);` 之类的函数

例如：

```
lv_obj_set_style_local_bg_color(slider, lv_color_red(), LV_PART_INDICATOR | LV_STATE_
↪FOCUSSED);
```

5.3.8 Properties (属性)

For the full list of style properties click [here](#).

有关样式属性的完整列表，请单击[此处](#) 查看。

Typical background properties (典型的背景属性)

In the documentation of the widgets you will see sentences like "The widget use the typical background properties". The "typical background properties" are the ones related to:

- Background
- Border
- Outline
- Shadow

- Padding
- Width and height transformation
- X and Y translation

在部件(widgets)的文档中，您将看到类似“小部件使用典型背景属性”这样的句子。“典型的背景属性”与以下相关：

- 背景
- 边境
- 大纲
- 阴影
- 填充
- 宽度和高度变换
- X 和 Y 平移

5.3.9 Transitions (过渡特效)

By default, when an object changes state (e.g. it's pressed) the new properties from the new state are set immediately. However, with transitions it's possible to play an animation on state change. For example, on pressing a button its background color can be animated to the pressed color over 300 ms.

The parameters of the transitions are stored in the styles. It's possible to set

- the time of the transition
- the delay before starting the transition
- the animation path (also known as timing or easing function)
- the properties to animate

The transition properties can be defined for each state. For example, setting 500 ms transition time in default state will mean that when the object goes to the default state a 500 ms transition time will be applied. Setting 100 ms transition time in the pressed state will mean a 100 ms transition time when going to pressed state. So this example configuration will result in going to pressed state quickly and then going back to default slowly.

To describe a transition an `lv_transition_dsc_t` variable needs to initialized and added to a style:

默认情况下，当一个对象改变状态（例如它被按下）时，新状态的新属性会立即设置。但是，通过转换，可以在状态更改时播放动画。例如，按下按钮时，其背景颜色可以在 300 毫秒内动画显示为按下的颜色。

过渡的参数存储在样式中。可以设置

- 过渡时期
- 开始过渡前的延迟
- 动画路径（也称为计时或缓动功能）
- 动画的属性

可以为每个状态定义转换属性。例如，在默认状态下设置 500 ms 转换时间意味着当对象进入默认状态时，将应用 500 ms 转换时间。在按下状态设置 100 ms 转换时间将意味着在进入按下状态时有 100 ms 转换时间。因此，此示例配置将导致快速进入按下状态，然后缓慢返回默认状态。

要描述转换，需要初始化 `lv_transition_dsc_t` 变量并将其添加到样式中：

```

/*Only its pointer is saved so must static, global or dynamically allocated */
static const lv_style_prop_t trans_props[] = {
    ↪ STYLE_BG_OPA, LV_STYLE_BG_COLOR,
    ↪ /*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);

```

LV_
0,

5.3.10 Color filter (色彩过滤)

TODO

5.3.11 Themes (主题)

Themes are a collection of styles. If there is an active theme LVGL applies it on every created widget. This will give a default appearance to the UI which can then be modified by adding further styles.

Every display can have a different theme. For example you could have a colorful theme on a TFT and monochrome theme on a secondary monochrome display.

To set a theme for a display, 2 steps are required:

1. Initialize a theme
2. Assign the initialized theme to a display.

Theme initialization functions can have different prototype. This example shows how to set the "default" theme:

主题是样式的集合。如果有一个活动主题 LVGL 将它应用到每个创建的小部件上。这将为 UI 提供默认外观，然后可以通过添加更多样式进行修改。

每个显示器都可以有不同的主题。例如，您可以在 TFT 上使用彩色主题，在辅助单色显示器上使用单色主题。

要为显示设置主题，需要 2 个步骤：

1. 初始化一个主题
2. 将初始化的主题分配给显示器。

主题初始化函数可以有不同的原型。此示例显示如何设置“默认”主题：

```

lv_theme_t * th = lv_theme_default_init(display, /*Use the DPI, size, etc from this
    ↪ display*/
                                         LV_COLOR_PALETTE_BLUE, LV_COLOR_PALETTE_CYAN,
                                         ↪ /*Primary and secondary palette*/
                                         false,      /*Light or dark mode*/
                                         &lv_font_montserrat_10, &lv_font_montserrat_
    ↪ 14, &lv_font_montserrat_18); /*Small, normal, large fonts*/

lv_disp_set_theme(display, th); /*Assign the theme to the display*/

```

The themes can be enabled in `lv_conf.h`. If the default theme is enabled by `LV_USE_THEME_DEFAULT 1` LVGL automatically initializes and sets it when a display is created.

可以在 `lv_conf.h` 中启用主题。如果默认主题由 `LV_USE_THEME_DEFAULT 1` 启用，LVGL 会在创建显示时自动初始化并设置它。

Extending themes (扩展主题)

Built-in themes can be extended. If a custom theme is created a parent theme can be selected. The parent theme's styles will be added before the custom theme's styles. Any number of themes can be chained this way. E.g. default theme -> custom theme -> dark theme.

`lv_theme_set_parent(new_theme, base_theme)` extends the `base_theme` with the `new_theme`.

There is an example for it below.

内置主题可以扩展。如果创建了自定义主题，则可以选择父主题。父主题的样式将添加在自定义主题的样式之前。可以通过这种方式链接任意数量的主题。例如。默认主题->自定义主题->深色主题。

`lv_theme_set_parent(new_theme, base_theme)` 使用 `new_theme` 扩展了 `base_theme`。

下面的是示例：

5.3.12 Examples

5.3.13 API

Typedefs

```
typedef uint8_t lv_blend_mode_t
typedef uint8_t lv_text_decor_t
typedef uint8_t lv_border_side_t
typedef uint8_t lv_grad_dir_t

typedef struct _lv_style_transiton_t lv_style_transition_dsc_t
    Descriptor for style transitions
```

Enums

enum [anonymous]

Possible options how to blend 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

enum [anonymous]

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 [anonymous]

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 [anonymous]

The direction of the gradient.

Values:

enumerator **LV_GRAD_DIR_NONE**

No gradient (the `grad_color` property is ignored)

enumerator **LV_GRAD_DIR_VER**

Vertical (top to bottom) gradient

enumerator **LV_GRAD_DIR_HOR**

Horizontal (left to right) gradient

enum `lv_style_prop_t`

Enumeration of all built in style properties

Values:

enumerator **LV_STYLE_PROP_INV**

enumerator **LV_STYLE_RADIUS**

enumerator **LV_STYLE_CLIP_CORNER**

enumerator **LV_STYLE_TRANSFORM_WIDTH**

enumerator **LV_STYLE_TRANSFORM_HEIGHT**

enumerator **LV_STYLE_TRANSLATE_X**

enumerator **LV_STYLE_TRANSLATE_Y**
enumerator **LV_STYLE_TRANSFORM_ZOOM**
enumerator **LV_STYLE_TRANSFORM_ANGLE**
enumerator **LV_STYLE_OPA**
enumerator **LV_STYLE_COLOR_FILTER_DSC**
enumerator **LV_STYLE_COLOR_FILTER_OPA**
enumerator **LV_STYLE_ANIM_TIME**
enumerator **LV_STYLE_ANIM_SPEED**
enumerator **LV_STYLE_TRANSITION**
enumerator **LV_STYLE_BLEND_MODE**
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_WIDTH**
enumerator **LV_STYLE_MIN_WIDTH**
enumerator **LV_STYLE_MAX_WIDTH**
enumerator **LV_STYLE_HEIGHT**
enumerator **LV_STYLE_MIN_HEIGHT**
enumerator **LV_STYLE_MAX_HEIGHT**
enumerator **LV_STYLE_X**
enumerator **LV_STYLE_Y**
enumerator **LV_STYLE_LAYOUT**
enumerator **LV_STYLE_ALIGN**
enumerator **LV_STYLE_BG_COLOR**
enumerator **LV_STYLE_BG_COLOR_FILTERED**
enumerator **LV_STYLE_BG_OPA**
enumerator **LV_STYLE_BG_GRAD_COLOR**
enumerator **LV_STYLE_BG_GRAD_COLOR_FILTERED**
enumerator **LV_STYLE_BG_GRAD_DIR**
enumerator **LV_STYLE_BG_MAIN_STOP**
enumerator **LV_STYLE_BG_GRAD_STOP**
enumerator **LV_STYLE_BG_IMG_SRC**
enumerator **LV_STYLE_BG_IMG_OPA**

enumerator `LV_STYLE_BG_IMG_RECOLOR`
enumerator `LV_STYLE_BG_IMG_RECOLOR_FILTERED`
enumerator `LV_STYLE_BG_IMG_RECOLOR_OPA`
enumerator `LV_STYLE_BG_IMG_TILED`
enumerator `LV_STYLE_BORDER_COLOR`
enumerator `LV_STYLE_BORDER_COLOR_FILTERED`
enumerator `LV_STYLE_BORDER_OPA`
enumerator `LV_STYLE_BORDER_WIDTH`
enumerator `LV_STYLE_BORDER_SIDE`
enumerator `LV_STYLE_BORDER_POST`
enumerator `LV_STYLE_OUTLINE_WIDTH`
enumerator `LV_STYLE_OUTLINE_COLOR`
enumerator `LV_STYLE_OUTLINE_COLOR_FILTERED`
enumerator `LV_STYLE_OUTLINE_OPA`
enumerator `LV_STYLE_OUTLINE_PAD`
enumerator `LV_STYLE_SHADOW_WIDTH`
enumerator `LV_STYLE_SHADOW_OFS_X`
enumerator `LV_STYLE_SHADOW_OFS_Y`
enumerator `LV_STYLE_SHADOW_SPREAD`
enumerator `LV_STYLE_SHADOW_COLOR`
enumerator `LV_STYLE_SHADOW_COLOR_FILTERED`
enumerator `LV_STYLE_SHADOW_OPA`
enumerator `LV_STYLE_IMG_OPA`
enumerator `LV_STYLE_IMG_RECOLOR`
enumerator `LV_STYLE_IMG_RECOLOR_FILTERED`
enumerator `LV_STYLE_IMG_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_COLOR_FILTERED`
enumerator `LV_STYLE_LINE_OPA`
enumerator `LV_STYLE_ARC_WIDTH`
enumerator `LV_STYLE_ARC_ROUNDED`
enumerator `LV_STYLE_ARC_COLOR`

```

enumerator LV_STYLE_ARC_COLOR_FILTERED
enumerator LV_STYLE_ARC_OPA
enumerator LV_STYLE_ARC_IMG_SRC
enumerator LV_STYLE_TEXT_COLOR
enumerator LV_STYLE_TEXT_COLOR_FILTERED
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_LAST_BUILT_IN_PROP
enumerator LV_STYLE_PROP_ANY

```

Functions

LV_EXPORT_CONST_INT(LV_IMG_ZOOM_NONE)

void **lv_style_init**(*lv_style_t* *style)
Initialize a style

注解: Do not call `lv_style_init` on styles that are already have some properties because this function won't free the used memory just set a default state for the style. In other words be sure to initialize styles only once!

参数 **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.

参数 **style** -- pointer to a style

lv_style_prop_t **lv_style_register_prop**(void)

bool **lv_style_remove_prop**(*lv_style_t* *style, *lv_style_prop_t* prop)
Remove a property from a style

参数

- **style** -- pointer to a style
- **prop** -- a style property ORed with a state.

返回 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()`

参数

- **style** -- pointer to style
- **prop** -- the ID of a property (e.g. LV_STLYE_BG_COLOR)
- **value** -- *lv_style_value_t* variable in which a filed is set according to the type of prop

`lv_res_t lv_style_get_prop(lv_style_t *style, lv_style_prop_t prop, lv_style_value_t *value)`

Get the value of a property

注解: For performance reasons there are no sanity check on **style**

参数

- **style** -- pointer to a style
- **prop** -- the ID of a property
- **value** -- pointer to a *lv_style_value_t* variable to store the value

返回 LV_RES_INV: the property wasn't found in the style (**value** is unchanged) LV_RES_OK: the property was found, and **value** is set accordingly

`static inline lv_res_t lv_style_get_prop_inlined(lv_style_t *style, lv_style_prop_t prop, lv_style_value_t *value)`

Get the value of a property

注解: For performance reasons there are no sanity check on **style**

注解: This function is the same as `lv_style_get_prop` but inlined. Use it only on performance critical places

参数

- **style** -- pointer to a style
- **prop** -- the ID of a property
- **value** -- pointer to a *lv_style_value_t* variable to store the value

返回 LV_RES_INV: the property wasn't found in the style (**value** is unchanged) LV_RES_OK: the property was found, 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)`

`lv_style_value_t lv_style_prop_get_default(lv_style_prop_t prop)`

Get the default value of a property

参数 **prop** -- the ID of a property

返回 the default value

`bool lv_style_is_empty(const lv_style_t *style)`

Checks if a style is empty (has no properties)

参数 **style** -- pointer to a style

返回

`uint8_t lv_style_get_prop_group(lv_style_prop_t prop)`

Tell the group of a property. If 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.

参数 **prop** -- a style property

返回 the group [0..7] 7 means all the custom properties with index > 112

`static inline void lv_style_set_pad_all(lv_style_t *style, lv_coord_t value)`

`static inline void lv_style_set_pad_hor(lv_style_t *style, lv_coord_t value)`

`static inline void lv_style_set_pad_ver(lv_style_t *style, lv_coord_t value)`

`static inline void lv_style_set_pad_gap(lv_style_t *style, lv_coord_t value)`

`static inline void lv_style_set_size(lv_style_t *style, lv_coord_t value)`

`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_transiton_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`

`lv_style_value_t value1`

`uint8_t *values_and_props`

`const lv_style_const_prop_t *const_props`

`union lv_style_t::[anonymous] v_p`

`uint16_t prop1`

`uint16_t is_const`

`uint8_t has_group`

`uint8_t prop_cnt`

Typedefs

```
typedef void (*lv_theme_apply_cb_t)(struct _lv_theme_t*, lv_obj_t*)
typedef struct _lv_theme_t lv_theme_t
```

Functions

`lv_theme_t *lv_theme_get_from_obj(lv_obj_t *obj)`

Get the theme assigned to the display of the object

参数 **obj** -- pointer to object

返回 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

参数 **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.

参数

- **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

参数

- **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

返回 pointer to the font

`const lv_font_t *lv_theme_get_font_normal(lv_obj_t *obj)`

Get the normal font of the theme

返回 pointer to the font

`const lv_font_t *lv_theme_get_font_large(lv_obj_t *obj)`

Get the subtitle font of the theme

返回 pointer to the font

`lv_color_t lv_theme_get_color_primary(lv_obj_t *obj)`

Get the primary color of the theme

返回 the color

`lv_color_t lv_theme_get_color_secondary(lv_obj_t *obj)`

Get the secondary color of the theme

返回 the color

```
struct _lv_theme_t
```

Public Members

```
lv_theme_apply_cb_t apply_cb
struct _lv_theme_t *parent
    Apply the current theme's style on top of this theme.

void *user_data
struct _lv_disp_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
```

Functions

```
static inline lv_coord_t lv_obj_get_style_radius(const struct _lv_obj_t *obj, uint32_t part)

static inline bool lv_obj_get_style_clip_corner(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_transform_width(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_transform_height(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_translate_x(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_translate_y(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_transform_zoom(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_transform_angle(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_opa_t lv_obj_get_style_opa(const struct _lv_obj_t *obj, uint32_t part)

static inline const lv_color_filter_dsc_t *lv_obj_get_style_color_filter_dsc(const struct _lv_obj_t *obj,
    uint32_t part)

static inline lv_opa_t lv_obj_get_style_color_filter_opa(const struct _lv_obj_t *obj, uint32_t part)
```

```
static inline uint32_t lv_obj_get_style_anim_time(const struct _lv_obj_t *obj, uint32_t part)

static inline uint32_t lv_obj_get_style_anim_speed(const struct _lv_obj_t *obj, uint32_t part)

static inline const lv_style_transition_dsc_t *lv_obj_get_style_transition(const struct _lv_obj_t *obj,
                                                                     uint32_t part)

static inline lv_blend_mode_t lv_obj_get_style_blend_mode(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_pad_top(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_pad_bottom(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_pad_left(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_pad_right(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_pad_row(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_pad_column(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_width(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_min_width(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_max_width(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_height(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_min_height(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_max_height(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_x(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_y(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_align_t lv_obj_get_style_align(const struct _lv_obj_t *obj, uint32_t part)

static inline uint16_t lv_obj_get_style_layout(const struct _lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_bg_color(const struct _lv_obj_t *obj, uint32_t part)
```

```

static inline lv_color_t lv_obj_get_style_bg_color_filtered(const struct lv_obj_t *obj, uint32_t part)

static inline lv_opa_t lv_obj_get_style_bg_opa(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_bg_grad_color(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_bg_grad_color_filtered(const struct lv_obj_t *obj,
                                                               uint32_t part)

static inline lv_grad_dir_t lv_obj_get_style_bg_grad_dir(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_bg_main_stop(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_bg_grad_stop(const struct lv_obj_t *obj, uint32_t part)

static inline const void *lv_obj_get_style_bg_img_src(const struct lv_obj_t *obj, uint32_t part)

static inline lv_opa_t lv_obj_get_style_bg_img_opa(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_bg_img_recolor(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_bg_img_recolor_filtered(const struct lv_obj_t *obj,
                                                               uint32_t part)

static inline lv_opa_t lv_obj_get_style_bg_img_recolor_opa(const struct lv_obj_t *obj, uint32_t part)

static inline bool lv_obj_get_style_bg_img_tiled(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_border_color(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_border_color_filtered(const struct lv_obj_t *obj, uint32_t
                                                               part)

static inline lv_opa_t lv_obj_get_style_border_opa(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_border_width(const struct lv_obj_t *obj, uint32_t part)

static inline lv_border_side_t lv_obj_get_style_border_side(const struct lv_obj_t *obj, uint32_t part)

static inline bool lv_obj_get_style_border_post(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_text_color(const struct lv_obj_t *obj, uint32_t part)

```

```
static inline lv_color_t lv_obj_get_style_text_color_filtered(const struct lv_obj_t *obj, uint32_t part)

static inline lv_opa_t lv_obj_get_style_text_opa(const struct lv_obj_t *obj, uint32_t part)

static inline const lv_font_t *lv_obj_get_style_text_font(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_text_letter_space(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_text_line_space(const struct lv_obj_t *obj, uint32_t part)

static inline lv_text_decor_t lv_obj_get_style_text_decor(const struct lv_obj_t *obj, uint32_t part)

static inline lv_text_align_t lv_obj_get_style_text_align(const struct lv_obj_t *obj, uint32_t part)

static inline lv_opa_t lv_obj_get_style_img_opa(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_img_recolor(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_img_recolor_filtered(const struct lv_obj_t *obj, uint32_t part)

static inline lv_opa_t lv_obj_get_style_img_recolor_opa(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_outline_width(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_outline_color(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_outline_color_filtered(const struct lv_obj_t *obj, uint32_t part)

static inline lv_opa_t lv_obj_get_style_outline_opa(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_outline_pad(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_shadow_width(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_shadow_ofs_x(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_shadow_ofs_y(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_shadow_spread(const struct lv_obj_t *obj, uint32_t part)
```

```

static inline lv_color_t lv_obj_get_style_shadow_color(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_shadow_color_filtered(const struct lv_obj_t *obj, uint32_t
                                                               part)

static inline lv_opa_t lv_obj_get_style_shadow_opa(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_line_width(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_line_dash_width(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_line_dash_gap(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_line_rounded(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_line_color(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_line_color_filtered(const struct lv_obj_t *obj, uint32_t
                                                               part)

static inline lv_opa_t lv_obj_get_style_line_opa(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_arc_width(const struct lv_obj_t *obj, uint32_t part)

static inline lv_coord_t lv_obj_get_style_arc_rounded(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_arc_color(const struct lv_obj_t *obj, uint32_t part)

static inline lv_color_t lv_obj_get_style_arc_color_filtered(const struct lv_obj_t *obj, uint32_t
                                                               part)

static inline lv_opa_t lv_obj_get_style_arc_opa(const struct lv_obj_t *obj, uint32_t part)

static inline const void *lv_obj_get_style_arc_img_src(const struct lv_obj_t *obj, uint32_t part)

static inline void lv_obj_set_style_radius(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t
                                         selector)

static inline void lv_obj_set_style_clip_corner(struct lv_obj_t *obj, bool value, lv_style_selector_t
                                                 selector)

static inline void lv_obj_set_style_transform_width(struct lv_obj_t *obj, lv_coord_t value,
                                                 lv_style_selector_t selector)

```

```

static inline void lv_obj_set_style_transform_height(struct lv_obj_t *obj, lv_coord_t value,
                                                lv_style_selector_t selector)

static inline void lv_obj_set_style_translate_x(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t
                                                selector)

static inline void lv_obj_set_style_translate_y(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t
                                                selector)

static inline void lv_obj_set_style_transform_zoom(struct lv_obj_t *obj, lv_coord_t value,
                                                lv_style_selector_t selector)

static inline void lv_obj_set_style_transform_angle(struct lv_obj_t *obj, lv_coord_t value,
                                                lv_style_selector_t selector)

static inline void lv_obj_set_style_opa(struct lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_color_filter_dsc(struct lv_obj_t *obj, const lv_color_filter_desc_t
                                                *value, lv_style_selector_t selector)

static inline void lv_obj_set_style_color_filter_opa(struct lv_obj_t *obj, lv_opa_t value,
                                                lv_style_selector_t selector)

static inline void lv_obj_set_style_anim_time(struct lv_obj_t *obj, uint32_t value, lv_style_selector_t
                                                selector)

static inline void lv_obj_set_style_anim_speed(struct lv_obj_t *obj, uint32_t value, lv_style_selector_t
                                                selector)

static inline void lv_obj_set_style_transition(struct lv_obj_t *obj, const lv_style_transition_desc_t *value,
                                                lv_style_selector_t selector)

static inline void lv_obj_set_style_blend_mode(struct lv_obj_t *obj, lv_blend_mode_t value,
                                                lv_style_selector_t selector)

static inline void lv_obj_set_style_pad_top(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t
                                                selector)

static inline void lv_obj_set_style_pad_bottom(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t
                                                selector)

static inline void lv_obj_set_style_pad_left(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t
                                                selector)

```

```

static inline void lv_obj_set_style_pad_right(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_pad_row(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_pad_column(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_width(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_min_width(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_max_width(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_height(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_min_height(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_max_height(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_x(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_y(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_align(struct lv_obj_t *obj, lv_align_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_layout(struct lv_obj_t *obj, uint16_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_bg_color(struct lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_bg_color_filtered(struct lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_bg_opa(struct lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_bg_grad_color(struct lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)

```

```

static inline void lv_obj_set_style_bg_grad_color_filtered(struct lv_obj_t *obj, lv_color_t value,
                                                       lv_style_selector_t selector)

static inline void lv_obj_set_style_bg_grad_dir(struct lv_obj_t *obj, lv_grad_dir_t value,
                                                lv_style_selector_t selector)

static inline void lv_obj_set_style_bg_main_stop(struct lv_obj_t *obj, lv_coord_t value,
                                                 lv_style_selector_t selector)

static inline void lv_obj_set_style_bg_grad_stop(struct lv_obj_t *obj, lv_coord_t value,
                                                lv_style_selector_t selector)

static inline void lv_obj_set_style_bg_img_src(struct lv_obj_t *obj, const void *value, lv_style_selector_t
                                               selector)

static inline void lv_obj_set_style_bg_img_opa(struct lv_obj_t *obj, lv_opa_t value, lv_style_selector_t
                                               selector)

static inline void lv_obj_set_style_bg_img_recolor(struct lv_obj_t *obj, lv_color_t value,
                                                 lv_style_selector_t selector)

static inline void lv_obj_set_style_bg_img_recolor_filtered(struct lv_obj_t *obj, lv_color_t value,
                                                          lv_style_selector_t selector)

static inline void lv_obj_set_style_bg_img_recolor_opa(struct lv_obj_t *obj, lv_opa_t value,
                                                       lv_style_selector_t selector)

static inline void lv_obj_set_style_bg_img_tiled(struct lv_obj_t *obj, bool value, lv_style_selector_t
                                                 selector)

static inline void lv_obj_set_style_border_color(struct lv_obj_t *obj, lv_color_t value,
                                                lv_style_selector_t selector)

static inline void lv_obj_set_style_border_color_filtered(struct lv_obj_t *obj, lv_color_t value,
                                                          lv_style_selector_t selector)

static inline void lv_obj_set_style_border_opa(struct lv_obj_t *obj, lv_opa_t value, lv_style_selector_t
                                               selector)

static inline void lv_obj_set_style_border_width(struct lv_obj_t *obj, lv_coord_t value,
                                                lv_style_selector_t selector)

static inline void lv_obj_set_style_border_side(struct lv_obj_t *obj, lv_border_side_t value,
                                                lv_style_selector_t selector)

```

```
static inline void lv_obj_set_style_border_post(struct lv_obj_t *obj, bool value, lv_style_selector_t selector)

static inline void lv_obj_set_style_text_color(struct lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_text_color_filtered(struct lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_text_opa(struct lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_text_font(struct lv_obj_t *obj, const lv_font_t *value, lv_style_selector_t selector)

static inline void lv_obj_set_style_text_letter_space(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_text_line_space(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_text_decor(struct lv_obj_t *obj, lv_text_decor_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_text_align(struct lv_obj_t *obj, lv_text_align_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_img_opa(struct lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_img_recolor(struct lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_img_recolor_filtered(struct lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_img_recolor_opa(struct lv_obj_t *obj, lv_opa_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_outline_width(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_outline_color(struct lv_obj_t *obj, lv_color_t value, lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_outline_color_filtered(struct lv_obj_t *obj, lv_color_t value,  
                                         lv_style_selector_t selector)  
  
static inline void lv_obj_set_style_outline_opa(struct lv_obj_t *obj, lv_opa_t value, lv_style_selector_t  
                                         selector)  
  
static inline void lv_obj_set_style_outline_pad(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t  
                                         selector)  
  
static inline void lv_obj_set_style_shadow_width(struct lv_obj_t *obj, lv_coord_t value,  
                                         lv_style_selector_t selector)  
  
static inline void lv_obj_set_style_shadow_ofs_x(struct lv_obj_t *obj, lv_coord_t value,  
                                         lv_style_selector_t selector)  
  
static inline void lv_obj_set_style_shadow_ofs_y(struct lv_obj_t *obj, lv_coord_t value,  
                                         lv_style_selector_t selector)  
  
static inline void lv_obj_set_style_shadow_spread(struct lv_obj_t *obj, lv_coord_t value,  
                                         lv_style_selector_t selector)  
  
static inline void lv_obj_set_style_shadow_color(struct lv_obj_t *obj, lv_color_t value,  
                                         lv_style_selector_t selector)  
  
static inline void lv_obj_set_style_shadow_color_filtered(struct lv_obj_t *obj, lv_color_t value,  
                                         lv_style_selector_t selector)  
  
static inline void lv_obj_set_style_shadow_opa(struct lv_obj_t *obj, lv_opa_t value, lv_style_selector_t  
                                         selector)  
  
static inline void lv_obj_set_style_line_width(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t  
                                         selector)  
  
static inline void lv_obj_set_style_line_dash_width(struct lv_obj_t *obj, lv_coord_t value,  
                                         lv_style_selector_t selector)  
  
static inline void lv_obj_set_style_line_dash_gap(struct lv_obj_t *obj, lv_coord_t value,  
                                         lv_style_selector_t selector)  
  
static inline void lv_obj_set_style_line_rounded(struct lv_obj_t *obj, lv_coord_t value,  
                                         lv_style_selector_t selector)  
  
static inline void lv_obj_set_style_line_color(struct lv_obj_t *obj, lv_color_t value, lv_style_selector_t  
                                         selector)
```

```

static inline void lv_obj_set_style_line_color_filtered(struct lv_obj_t *obj, lv_color_t value,
                                                       lv_style_selector_t selector)

static inline void lv_obj_set_style_line_opa(struct lv_obj_t *obj, lv_opa_t value, lv_style_selector_t
                                              selector)

static inline void lv_obj_set_style_arc_width(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t
                                              selector)

static inline void lv_obj_set_style_arc_rounded(struct lv_obj_t *obj, lv_coord_t value, lv_style_selector_t
                                              selector)

static inline void lv_obj_set_style_arc_color(struct lv_obj_t *obj, lv_color_t value, lv_style_selector_t
                                              selector)

static inline void lv_obj_set_style_arc_color_filtered(struct lv_obj_t *obj, lv_color_t value,
                                                       lv_style_selector_t selector)

static inline void lv_obj_set_style_arc_opa(struct lv_obj_t *obj, lv_opa_t value, lv_style_selector_t
                                              selector)

static inline void lv_obj_set_style_arc_img_src(struct lv_obj_t *obj, const void *value,
                                                lv_style_selector_t selector)

```

Functions

```

static inline void lv_style_set_radius(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_clip_corner(lv_style_t *style, bool value)

static inline void lv_style_set_transform_width(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_transform_height(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_translate_x(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_translate_y(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_transform_zoom(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_transform_angle(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_opa(lv_style_t *style, lv_opa_t value)

```

```
static inline void lv_style_set_color_filter_dsc(lv_style_t *style, const lv_color_filter_dsc_t *value)

static inline void lv_style_set_color_filter_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_anim_time(lv_style_t *style, uint32_t value)

static inline void lv_style_set_anim_speed(lv_style_t *style, uint32_t value)

static inline void lv_style_set_transition(lv_style_t *style, const lv_style_transition_dsc_t *value)

static inline void lv_style_set_blend_mode(lv_style_t *style, lv_blend_mode_t value)

static inline void lv_style_set_pad_top(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_pad_bottom(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_pad_left(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_pad_right(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_pad_row(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_pad_column(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_width(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_min_width(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_max_width(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_height(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_min_height(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_max_height(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_x(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_y(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_align(lv_style_t *style, lv_align_t value)
```

```
static inline void lv_style_set_layout(lv_style_t *style, uint16_t value)

static inline void lv_style_set_bg_color(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_bg_color_filtered(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_bg_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_bg_grad_color(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_bg_grad_color_filtered(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_bg_grad_dir(lv_style_t *style, lv_grad_dir_t value)

static inline void lv_style_set_bg_main_stop(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_bg_grad_stop(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_bg_img_src(lv_style_t *style, const void *value)

static inline void lv_style_set_bg_img_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_bg_img_recolor(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_bg_img_recolor_filtered(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_bg_img_recolor_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_bg_img_tiled(lv_style_t *style, bool value)

static inline void lv_style_set_border_color(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_border_color_filtered(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_border_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_border_width(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_border_side(lv_style_t *style, lv_border_side_t value)

static inline void lv_style_set_border_post(lv_style_t *style, bool value)
```

```
static inline void lv_style_set_text_color(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_text_color_filtered(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_text_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_text_font(lv_style_t *style, const lv_font_t *value)

static inline void lv_style_set_text_letter_space(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_text_line_space(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_text_decor(lv_style_t *style, lv_text_decor_t value)

static inline void lv_style_set_text_align(lv_style_t *style, lv_text_align_t value)

static inline void lv_style_set_img_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_img_recolor(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_img_recolor_filtered(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_img_recolor_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_outline_width(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_outline_color(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_outline_color_filtered(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_outline_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_outline_pad(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_shadow_width(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_shadow_ofs_x(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_shadow_ofs_y(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_shadow_spread(lv_style_t *style, lv_coord_t value)
```

```
static inline void lv_style_set_shadow_color(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_shadow_color_filtered(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_shadow_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_line_width(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_line_dash_width(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_line_dash_gap(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_line_rounded(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_line_color(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_line_color_filtered(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_line_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_arc_width(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_arc_rounded(lv_style_t *style, lv_coord_t value)

static inline void lv_style_set_arc_color(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_arc_color_filtered(lv_style_t *style, lv_color_t value)

static inline void lv_style_set_arc_opa(lv_style_t *style, lv_opa_t value)

static inline void lv_style_set_arc_img_src(lv_style_t *style, const void *value)
```

5.4 Style properties

5.4.1 Size and position

TODO

width

Sets the width of object. Pixel, percentage and LV_SIZE_CONTENT values can be used. Percentage values are relative to the width of the parent's content area.

min_width

Sets a minimal width. Pixel and percentage values can be used. Percentage values are relative to the width of the parent's content area.

max_width

Sets a maximal width. Pixel and percentage values can be used. Percentage values are relative to the width of the parent's content area.

height

Sets the height of object. Pixel, percentage and LV_SIZE_CONTENT can be used. Percentage values are relative to the height of the parent's content area.

min_height

Sets a minimal height. Pixel and percentage values can be used. Percentage values are relative to the width of the parent's content area.

max_height

Sets a maximal height. Pixel and percentage values can be used. Percentage values are relative to the height of the parent's content area.

x

Set the X coordinate of the object considering the set `align`. Pixel and percentage values can be used. Percentage values are relative to the width of the parent's content area.

y

Set the Y coordinate of the object considering the set `align`. Pixel and percentage values can be used. Percentage values are relative to the height of the parent's content area.

align

Set the alignment which determines from which point of the parent the X and Y coordinates should be interpreted. The possible values are: LV_ALIGN_TOP_LEFT/MID/RIGHT, LV_ALIGN_BOTTOM_LEFT/MID/RIGHT, LV_ALIGN_LEFT/RIGHT_MID, LV_ALIGN_CENTER

transform_width

Make the object wider on both sides with this value. Pixel and percentage (with `lv_pct(x)`) values can be used. Percentage values are relative to the object's width.

transform_height

Make the object higher on both sides with this value. Pixel and percentage (with `lv_pct(x)`) values can be used. Percentage values are relative to the object's height.

translate_x

Move the object with this value in X direction. Applied after layouts, aligns and other positioning. Pixel and percentage (with `lv_pct(x)`) values can be used. Percentage values are relative to the object's width.

translate_y

Move the object with this value in Y direction. Applied after layouts, aligns and other positioning. Pixel and percentage (with `lv_pct(x)`) values can be used. Percentage values are relative to the object's height.

transform_zoom

Zoom image-like objects. Multiplied with the zoom set on the object. The value 256 (or LV_IMG_ZOOM_NONE) means normal size, 128 half size, 512 double size, and so on

transform_angle

Rotate image-like objects. Added to the rotation set on the object. The value is interpreted in 0.1 degree unit. E.g. 45 deg. = 450

5.4.2 Padding

TODO

pad_top

Sets the padding on the top. It makes the content area smaller in this direction.

pad_bottom

Sets the padding on the bottom. It makes the content area smaller in this direction.

pad_left

Sets the padding on the left. It makes the content area smaller in this direction.

pad_right

Sets the padding on the right. It makes the content area smaller in this direction.

pad_row

Sets the padding between the rows. Used by the layouts.

pad_column

Sets the padding between the columns. Used by the layouts.

5.4.3 Miscellaneous

TODO

radius

Set the radius on every corner. The value is interpreted in pixel (≥ 0) or `LV_RADIUS_CIRCLE` for max. radius

clip_corner

Enable to clip the overflowed content on the rounded corner. Can be `true` or `false`.

opa

Scale down all opacity values of the object by this factor. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or `LV_OPA_10`, `LV_OPA_20`, etc indicate semi-transparency.

color_filter_dsc

Mix a color to all colors of the object.

color_filter_opa

The intensity of mixing of color filter.

anim_time

The animation time in milliseconds. Its meaning is widget specific. E.g. blink time of the cursor on the text area or scroll time of a roller. See the widgets' documentation to learn more.

anim_speed

The animation speed in pixel/sec. Its meaning is widget specific. E.g. scroll speed of label. See the widgets' documentation to learn more.

transition

An initialized `lv_style_transition_dsc_t` to describe a transition.

blend_mode

Describes how to blend the colors to the background. The possible values are `LV_BLEND_MODE_NORMAL/ADDITION/SUBTRACTIVE`

layout

Set the layout if the object. The children will be repositioned and resized according to the policies set for the layout. For the possible values see the documentation of the layouts.

base_dir

Set the base direction of the object. The possible values are `LV_BIDI_DIR_LTR/RTL/AUTO`.

5.4.4 Background

TODO

bg_color

Set the background color of the object.

bg_opa

Set the opacity of the background. Value 0, LV_OPA_0 or LV_OPA_TRANSP means fully transparent, 256, LV_OPA_100 or LV_OPA_COVER means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

bg_grad_color

Set the gradient color of the background. Used only if `grad_dir` is not LV_GRAD_DIR_NONE

bg_grad_dir

Set the direction of the gradient of the background. The possible values are LV_GRAD_DIR_NONE/HOR/VER.

bg_main_stop

Set the point from which the background color should start for gradients. 0 means to top/left side, 255 the bottom/right side, 128 the center, and so on

bg_grad_stop

Set the point from which the background's gradient color should start. 0 means to top/left side, 255 the bottom/right side, 128 the center, and so on

bg_img_src

Set a background image. Can be a pointer to `lv_img_dsc_t`, a path to a file or an LV_SYMBOL_...

bg_img_opa

Set the opacity of the background image. Value 0, LV_OPA_0 or LV_OPA_TRANSP means fully transparent, 256, LV_OPA_100 or LV_OPA_COVER means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

bg_img_recolor

Set a color to mix to the background image.

bg_img_recolor_opa

Set the intensity of background image recoloring. Value 0, LV_OPA_0 or LV_OPA_TRANSPI means no mixing, 256, LV_OPA_100 or LV_OPA_COVER means full recoloring, other values or LV_OPA_10, LV_OPA_20, etc are interpreted proportionally.

bg_img_tiled

If enabled the background image will be tiled. The possible values are `true` or `false`.

5.4.5 Border

TODO

border_color

Set the color of the border

border_opa

Set the opacity of the border. Value 0, LV_OPA_0 or LV_OPA_TRANSPI means fully transparent, 256, LV_OPA_100 or LV_OPA_COVER means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

border_width

Set the width of the border. Only pixel values can be used.

border_side

Set which side(s) the border should be drawn. The possible values are LV_BORDER_SIDE_NONE/TOP/BOTTOM/LEFT/RIGHT/INTERNAL. OR-ed values can be used as well, e.g. LV_BORDER_SIDE_TOP | LV_BORDER_SIDE_LEFT.

border_post

Sets whether the border should be drawn before or after the children are drawn. `true`: after children, `false`: before children

5.4.6 Text

TODO

text_color

Sets the color of the text.

text_opa

Set the opacity of the text. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or `LV_OPA_10`, `LV_OPA_20`, etc indicate semi-transparency.

text_font

Set the font of the text (a pointer `lv_font_t *`).

text_letter_space

Set the letter space in pixels

text_line_space

Set the line space in pixels.

text_decor

Set decoration for the text. The possible values are `LV_TEXT_DECOR_NONE`/`UNDERLINE`/`STRIKETHROUGH`. OR-ed values can be used as well.

text_align

Set how to align the lines of the text. Note that it doesn't align the object itself, only the lines inside the object. The possible values are `LV_TEXT_ALIGN_LEFT`/`CENTER`/`RIGHT`/`AUTO`. `LV_TEXT_ALIGN_AUTO` detect the text base direction and uses left or right alignment accordingly

5.4.7 Image

TODO

img_opa

Set the opacity of an image. Value 0, `LV_OPA_0` or `LV_OPA_TRANSP` means fully transparent, 256, `LV_OPA_100` or `LV_OPA_COVER` means fully covering, other values or `LV_OPA_10`, `LV_OPA_20`, etc indicate semi-transparency.

img_recolor

Set color to mix to the image.

img_recolor_opa

Set the intensity of the color mixing. Value 0, LV_OPA_0 or LV_OPA_TRANSP means fully transparent, 256, LV_OPA_100 or LV_OPA_COVER means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

5.4.8 Outline

TODO

outline_width

Set the width of the outline in pixels.

outline_color

Set the color of the outline.

outline_opa

Set the opacity of the outline. Value 0, LV_OPA_0 or LV_OPA_TRANSP means fully transparent, 256, LV_OPA_100 or LV_OPA_COVER means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

outline_pad

Set the padding of the outline, i.e. the gap between object and the outline.

5.4.9 Shadow

TODO

shadow_width

Set the width of the shadow in pixels. The value should be ≥ 0 .

shadow_ofs_x

Set an offset on the shadow in pixels in X direction.

shadow_ofs_y

Set an offset on the shadow in pixels in Y direction.

shadow_spread

Make the shadow calculation to use a larger or smaller rectangle as base. The value can be in pixel to make the area larger/smaller

shadow_color

Set the color of the shadow

shadow_opa

Set the opacity of the shadow. Value 0, LV_OPA_0 or LV_OPA_TRANSP means fully transparent, 256, LV_OPA_100 or LV_OPA_COVER means fully covering, other values or LV_OPA_10, LV_OPA_20, etc indicate semi-transparency.

5.4.10 Line

TODO

line_width

Set the width of the lines in pixel.

line_dash_width

Set the width of dashes in pixel. Note that dash works only on horizontal and vertical lines

line_dash_gap

Set the gap between dashes in pixel. Note that dash works only on horizontal and vertical lines

line_rounded

Make the end points of the lines rounded. `true`: rounded, `false`: perpendicular line ending

line_color

Set the color fo the lines.

line_opa

Set the opacity of the lines.

5.4.11 Arc

TODO

arc_width

Set the width (thickness) of the arcs in pixel.

arc_rounded

Make the end points of the arcs rounded. `true`: rounded, `false`: perpendicular line ending

arc_color

Set the color of the arc.

arc_opa

Set the opacity of the arcs.

arc_img_src

Set an image from which the arc will be masked out. It's useful to display complex effects on the arcs. Can be a pointer to `lv_img_dsc_t` or a path to a file

5.5 Scroll (滚动)**5.5.1 Overview (概述)**

In LVGL scrolling works very intuitively: if an object is out of its parent content area (the size without paddings), the parent becomes scrollable and scrollbar(s) will appear. That's it.

Any object can be scrollable including `lv_obj_t`, `lv_img`, `lv_btn`, `lv_meter`, etc

The object can either be scrolled either horizontally or vertically in one stroke; diagonal scrolling is not possible.

在 LVGL 中，滚动的工作非常直观：如果对象超出其父内容区域（没有填充的大小），则父对象的空间可滚动并且会出现滚动条。仅此而已。

任何对象都可以滚动，包括 `lv_obj_t`、`lv_img`、`lv_btn`、`lv_meter` 等

对象可以一次水平或垂直滚动；对角滚动是不可能的。

Scrollbar (滚动条)

Mode (模式)

The scrollbars are displayed according to the set `mode`. The following `modes` exist:

- `LV_SCROLLBAR_MODE_OFF` Never show the scrollbars
- `LV_SCROLLBAR_MODE_ON` Always show the scrollbars
- `LV_SCROLLBAR_MODE_ACTIVE` Show scroll bars while object is being scrolled
- `LV_SCROLLBAR_MODE_AUTO` Show scroll bars when the content is large enough to be scrolled

`lv_obj_set_scrollbar_mode(obj, LV_SCROLLBAR_MODE_...)` set the scrollbar mode on an object.

滚动条根据设置的 模式显示。有下面这几种 模式可以选择：

- `LV_SCROLLBAR_MODE_OFF` 从不显示滚动条
- `LV_SCROLLBAR_MODE_ON` 始终显示滚动条
- `LV_SCROLLBAR_MODE_ACTIVE` 在对象滚动时显示滚动条
- `LV_SCROLLBAR_MODE_AUTO` 当内容足够大可以滚动时显示滚动条

`lv_obj_set_scrollbar_mode(obj, LV_SCROLLBAR_MODE_...)` 在对象上设置滚动条模式。

Styling (样式)

The scrollbars have their own dedicated part, called `LV_PART_SCROLLBAR`. For example a scrollbar can turned to red like this:

滚动条有自己的专用部分，称为 `LV_PART_SCROLLBAR`。例如，滚动条可以像这样变成红色：

```
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(obj, &style_red, LV_PART_SCROLLBAR);
```

The object goes to `LV_STATE_SCROLLED` state while it's being scrolled. It allows adding different style to the scrollbar or the object itself when scrolled. This code makes the scrollbar blue when the object is scrolled:

对象在滚动时进入 `LV_STATE_SCROLLED` 状态。它允许在滚动时向滚动条或对象本身添加不同的样式。当对象滚动时，此代码使滚动条变为蓝色：

```

static lv_style_t style_blue;
lv_style_init(&style_blue);
lv_style_set_bg_color(&style_red, lv_color_blue());

...
lv_obj_add_style(obj, &style_blue, LV_STATE_SCROLLED | LV_PART_SCROLLBAR);

```

Events(事件)

The following events are related to scrolling:

- LV_EVENT_SCROLL_BEGIN Scrolling begins
- LV_EVENT_SCROLL_END Scrolling ends
- LV_EVENT_SCROLL Scroll happened. Triggered on every position change. Scroll events

以下事件与滚动相关:

- LV_EVENT_SCROLL_BEGIN 开始滚动
- LV_EVENT_SCROLL_END 滚动结束
- LV_EVENT_SCROLL 滚动发生。每次位置变化时触发。

滚动事件

5.5.2 Basic example (基本示例)

TODO

5.5.3 Features of scrolling (滚动的特点)

Besides managing "normal" scrolling there are many interesting and useful additional features too.

除了管理“正常”滚动之外，还有许多有趣且有用的附加功能。

Scollable (滚动效果)

It's possible to make an object non-scrollable with `lv_obj_clear_flag(obj, LV_OBJ_FLAG_SCROLLABLE)`.

Non-scrollable object can still propagate the scrolling (chain) to the parents.

The direction in which scrolling can happen can be controlled by `lv_obj_set_scroll_dir(obj, LV_DIR_...)`. The following values are possible for the direction:

- LV_DIR_TOP only scroll up
- LV_DIR_LEFT only scroll left
- LV_DIR_BOTTOM only scroll down
- LV_DIR_RIGHT only scroll right
- LV_DIR_HOR only scroll horizontally
- LV_DIR_TOP only scroll vertically

- `LV_DIR_ALL` scroll any directions

OR-ed values are also possible. E.g. `LV_DIR_TOP | LV_DIR_LEFT`.

可以使用 `lv_obj_clear_flag(obj, LV_OBJ_FLAG_SCROLLABLE)` 使对象不可滚动。

不可滚动对象仍然可以将滚动（链）传播到父对象。

滚动发生的方向可以由 `lv_obj_set_scroll_dir(obj, LV_DIR_...)` 控制。方向可能有以下值：

- `LV_DIR_TOP` 只向上滚动
- `LV_DIR_LEFT` 只向左滚动
- `LV_DIR_BOTTOM` 只向下滚动
- `LV_DIR_RIGHT` 只向右滚动
- `LV_DIR_HOR` 只能水平滚动
- `LV_DIR_TOP` 只能垂直滚动
- `LV_DIR_ALL` 滚动任何方向

可以用同时设置使用多个值 (OR-ed)。例如。`LV_DIR_TOP | LV_DIR_LEFT`。

Scroll chain (滚动条)

If an object can't be scrolled further (e.g. its content has reached the bottom most position) the scrolling is propagated to its parent. If the parent can be scrolled in that direction than it will be scrolled instead. It propagates to the grandparent and grand-grandparents too.

The propagation on scrolling is called "scroll chaining" and it can be enabled/disabled with the `LV_OBJ_FLAG_SCROLL_CHAIN` flag. If chaining is disabled the propagation stops on the object and the parent(s) won't be scrolled.

如果对象无法进一步滚动（例如，它的内容已到达最底部的位置），则滚动将传播到其父级。如果父级在该方向上滚动，则它将改为滚动。它也传播给祖父母和祖父母。

滚动传播称为“滚动链接”，可以使用 `LV_OBJ_FLAG_SCROLL_CHAIN` 标志启用/禁用。如果禁用链接，则传播将停止在对象上，并且不会滚动父对象。

Scroll momentum (滚动惯性效果)

When the user scrolls an object and releases it, LVGL can emulate a momentum for the scrolling. It's like the object was thrown and scrolling slows down smoothly.

The scroll momentum can be enabled/disabled with the `LV_OBJ_FLAG_SCROLL_MOMENTUM` flag.

当用户滚动对象并释放它时，LVGL 可以模拟滚动的动量。就好像对象被抛出并且滚动平稳地减慢了速度。

可以使用“`LV_OBJ_FLAG_SCROLL_MOMENTUM`”标志启用/禁用滚动动量。

Elastic scroll (弹性卷轴效果)

Normally the content can't be scrolled inside the object. That is the top side of the content can't be below the top side of the object.

However, with `LV_OBJ_FLAG_SCROLL_ELASTIC` a fancy effect can be added when the user "over-scrolls" the content. The scrolling slows down, and the content can be scrolled inside the object. When the object is released the content scrolled in it will be animated back to the valid position.

通常内容不能在对象内滚动。即内容的顶部不能低于对象的顶部。

但是，使用 `LV_OBJ_FLAG_SCROLL_ELASTIC` 可以在用户“滚动”内容时添加奇特的效果。滚动变慢，内容可以在对象内部滚动。当对象被释放时，滚动到其中的内容将动画回到有效位置。

Snapping (捕捉)

The children of an object can be snapped according to specific rules when scrolling ends. Children can be made snappable individually with the `LV_OBJ_FLAG_SNAPPABLE` flag.

The object can align the snapped children in 4 ways:

- `LV_SCROLL_SNAP_NONE` Snapping is disabled. (default)
- `LV_SCROLL_SNAP_START` Align the children to the left/top side of the scrolled object
- `LV_SCROLL_SNAP_END` Align the children to the right/bottom side of the scrolled object
- `LV_SCROLL_SNAP_CENTER` Align the children to the center of the scrolled object

滚动结束时，可以根据特定规则捕捉对象的子项。可以使用 `LV_OBJ_FLAG_SNAPPABLE` 标志将子对象单独设置为可捕捉。

该对象可以通过 4 种方式对齐对齐的子项：

- `LV_SCROLL_SNAP_NONE` 捕捉被禁用。(默认)
- `LV_SCROLL_SNAP_START` 将子对象与滚动对象的左侧/顶部对齐
- `LV_SCROLL_SNAP_END` 将子对象与滚动对象的右侧/底部对齐
- `LV_SCROLL_SNAP_CENTER` 将子对象与滚动对象的中心对齐

The alignment can be set with `lv_obj_set_scroll_snap_x/y(obj, LV_SCROLL_SNAP_...)`:

Under the hood the following happens:

1. User scrolls an object and releases the screen
2. LVGL calculates where the scroll would end considering scroll momentum
3. LVGL finds the nearest scroll point
4. LVGL scrolls to the snap point with an animation

对齐可以用 `lv_obj_set_scroll_snap_x/y(obj, LV_SCROLL_SNAP_...)` 设置：

底层代码会发生以下情况：

1. 用户滚动对象并释放屏幕
2. LVGL 考虑滚动动量计算滚动结束的位置
3. LVGL 寻找最近的滚动点
4. LVGL 滚动到带动画的捕捉点

Scroll one(只滚动一个)

The "scroll one" feature tells LVGL to allow scrolling only one snappable child at a time. So this requires to make the children snappable and set a scroll snap alignment different from `LV_SCROLL_SNAP_NONE`.

This feature can be enabled by the `LV_OBJ_FLAG_SCROLL_ONE` flag.

“只滚动一个 (Scroll one)” 功能告诉 LVGL 一次只允许滚动一个可捕捉的孩子。

因此，这需要使子项可捕捉并设置与 `LV_SCROLL_SNAP_NONE` 不同的滚动对齐对齐方式。此功能可以通过 `LV_OBJ_FLAG_SCROLL_ONE` 标志启用。

Scroll on focus (滚动焦点)

Imagine that there a lot of objects in a group that are on scrollable object. Pressing the "Tab" button focuses the next object but it might be out of the visible area of the scrollable object. If the "scroll on focus" features is enabled LVGL will automatically scroll to the objects to bring the children into the view. The scrolling happens recursively therefore even nested scrollable object are handled properly. The object will be scrolled to the view even if it's on a different page of a tabview.

想象一下，一个组中有很多对象位于可滚动对象上。按 “Tab” 按钮聚焦下一个对象，但它可能超出可滚动对象的可见区域。如果启用了 “焦点滚动” 功能，LVGL 将自动滚动到对象以将子项带入视图。

滚动以递归方式发生，因此即使嵌套的可滚动对象也能得到正确处理。即使对象位于 tabview 的不同页面上，它也会滚动到视图。

5.5.4 Scroll manually

The following API functions allow to manually scroll objects:

- `lv_obj_scroll_by(obj, x, y, LV_ANIM_ON/OFF)` scroll by x and y values
- `lv_obj_scroll_to(obj, x, y, LV_ANIM_ON/OFF)` scroll to bring the given coordinate to the top left corner
- `lv_obj_scroll_to_x(obj, x, LV_ANIM_ON/OFF)` scroll to bring the given coordinate to the left side
- `lv_obj_scroll_to_y(obj, y, LV_ANIM_ON/OFF)` scroll to bring the given coordinate to the left side

以下 API 函数允许手动滚动对象：

- `lv_obj_scroll_by(obj, x, y, LV_ANIM_ON/OFF)` 按 x 和 y 值滚动
- `lv_obj_scroll_to(obj, x, y, LV_ANIM_ON/OFF)` 滚动以将给定的坐标带到左上角
- `lv_obj_scroll_to_x(obj, x, LV_ANIM_ON/OFF)` 滚动以将给定的坐标带到左侧
- `lv_obj_scroll_to_y(obj, y, LV_ANIM_ON/OFF)` 滚动以将给定的坐标带到左侧

5.5.5 Self size (自身尺寸)

Self size is a property of an object. Normally, the user shouldn't use this parameter but if a custom widget is created it might be useful.

In short, self size tell the size of the content. To understand it better take the example of a table. Let's say it has 10 rows each with 50 px height. So the total height of the content is 500 px. In other words the "self height" is 500 px. If the user sets only 200 px height for the table LVGL will see that the self size is larger and make the table scrollable.

It means not only the children can make an object scrollable but a larger self size too.

LVGL uses the `LV_EVENT_GET_SELF_SIZE` event to get the self size of an object. Here is an example to see how to handle the event

自身大小是对象的属性。通常，用户不应使用此参数，但如果创建了自定义小、部件，它可能会很有用。

简而言之，自我大小告诉内容的大小。为了更好地理解它，举一个表格的例子。假设它有 10 行，每行 50 像素高度。所以内容的总高度是 500 px。换句话说，“自身高度”是 500 像素。如果用户只为表格设置 200 像素高度，LVGL 将看到自身尺寸更大并使表格可滚动。

这意味着不仅孩子们可以使对象可滚动，而且还可以使自身尺寸更大。

LVGL 使用 `LV_EVENT_GET_SELF_SIZE` 事件来获取对象的自身大小。下面是一个例子，看看如何处理事件

```
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 the self width
    }

    if(p->y >= 0) {
        p->y = 50;           //Set or calculate the self height
    }
}
```

5.5.6 Examples

5.6 Layers (图层)

5.6.1 Order of creation (图层顺序)

By default, LVGL draws new objects on top of old objects.

For example, assume we added a button to a parent object named button1 and then another button named button2. Then button1 (with its child object(s)) will be in the background and can be covered by button2 and its children.

默认情况下，LVGL 在旧对象之上绘制新对象。

例如，假设我们向名为 button1 的父对象添加了一个按钮，然后添加了另一个名为 button2 的按钮。然后 button1 (及其子对象) 将在背景中并且可以被 button2 及其子对象覆盖。



Button 1

```

/*Create a screen*/
lv_obj_t * scr = lv_obj_create(NULL, NULL);
lv_scr_load(scr);           /*Load the screen*/

/*Create 2 buttons*/
lv_obj_t * btn1 = lv_btn_create(scr, NULL);           /*Create a button on the screen*/
lv_btn_set_fit(btn1, true, true);                     /*Enable to automatically set the size according to the content*/
lv_obj_set_pos(btn1, 60, 40);                         /*Set the position of the button*/
                                                        /*size according to the content*/
                                                        /*button*/

lv_obj_t * btn2 = lv_btn_create(scr, btn1);           /*Copy the first button*/
lv_obj_set_pos(btn2, 180, 80);                        /*Set the position of the button*/

/*Add labels to the buttons*/
lv_obj_t * label1 = lv_label_create(btn1, NULL);       /*Create a label on the first button*/
lv_label_set_text(label1, "Button 1");                 /*Set the text of the label*/
                                                        /*label*/
                                                        /*second button*/
lv_label_set_text(label2, "Button 2");                 /*Create a label on the second button*/
                                                        /*label*/
                                                        /*Delete the second label*/
lv_obj_del(label2);

```

5.6.2 Bring to the foreground (前台展示)

There are 2 explicit way to bring an object to the foreground:

- Use `lv_obj_move_foreground(obj)` to explicitly tell the library to bring an object to the foreground. Similarly, use `lv_obj_move_background(obj)` to move to the background.
- When `lv_obj_set_parent(obj, new_parent)` is used, `obj` will be on the foreground on the `new_parent`.

有两种显式方法可以将对象置于前台：

- 使用 `lv_obj_move_foreground(obj)` 明确告诉库将一个对象带到前台。同样，使用 `lv_obj_move_background(obj)` 移动到背景。
- 当使用 `lv_obj_set_parent(obj, new_parent)` 时，`obj` 将位于 `new_parent` 的前台。

5.6.3 Top and sys layers (顶层和系统层)

LVGL uses two special layers named as `layer_top` and `layer_sys`. Both are visible and common on all screens of a display. **They are not, however, shared among multiple physical displays.** The `layer_top` is always on top of the default screen (`lv_scr_act()`), and `layer_sys` is on top of `layer_top`.

The `layer_top` can be used by the user to create some content visible everywhere. For example, a menu bar, a pop-up, etc. If the `click` attribute is enabled, then `layer_top` will absorb all user click and acts as a modal.

LVGL 使用名为“`layer_top`”和“`layer_sys`”的两个特殊层。两者在显示器的所有屏幕上都是可见的和通用的。然而，它们不会在多个物理显示器之间共享。`layer_top` 始终位于默认屏幕的顶部 (`lv_scr_act()`)，而 `layer_sys` 位于 `layer_top` 的顶部。

用户可以使用 `layer_top` 来创建一些随处可见的内容。例如，菜单栏、弹出窗口等。如果启用了 `click` 属性，那么 `layer_top` 将吸收所有用户点击并充当模态。

```
lv_obj_set_click(lv_layer_top(), true);
```

The `layer_sys` is also used for similar purposes on LVGL. For example, it places the mouse cursor above all layers to be sure it's always visible.

`layer_sys` 也用于 LVGL 的类似目的。例如，它将鼠标光标放在所有图层上方以确保它始终可见。

5.7 Events (事件)

Events are triggered in LVGL when something happens which might be interesting to the user, e.g. when an object

- is clicked
- is scrolled
- has its value changed
- is redrawn, etc.

当发生用户可能感兴趣的事情时，LVGL 中会触发事件，例如当一个对象

- 被点击
- 滚动
- 改变了它的价值

- 重绘等。

5.7.1 Add events to the object

The user can assign callback functions to an object to see its events. In practice, it looks like this:

用户可以为对象分配回调函数以查看其事件。在实践中，它看起来像这样：

```
lv_obj_t * btn = lv_btn_create(lv_scr_act());
lv_obj_add_event_cb(btn, my_event_cb, LV_EVENT_CLICKED, NULL); /*Assign an event
→callback*/
...
static void my_event_cb(lv_event_t * event)
{
    printf("Clicked\n");
}
```

In the example `LV_EVENT_CLICKED` means that only the click event will call `my_event_cb`. See the [list of event codes](#) for all the options. `LV_EVENT_ALL` can be used to receive all the events.

The last parameter of `lv_obj_add_event_cb` is a pointer to any custom data that will be available in the event. It will be described later in more detail.

More events can be added to an object, like this:

在示例中 `LV_EVENT_CLICKED` 意味着只有点击事件会调用 `my_event_cb`。有关所有选项，请参阅[事件代码列表](#)。`LV_EVENT_ALL` 可用于接收所有事件。

`lv_obj_add_event_cb` 的最后一个参数是指向事件中可用的任何自定义数据的指针。稍后将更详细地描述。

可以向一个对象添加更多事件，如下所示：

```
lv_obj_add_event_cb(obj, my_event_cb_1, LV_EVENT_CLICKED, NULL);
lv_obj_add_event_cb(obj, my_event_cb_2, LV_EVENT_PRESSED, NULL);
lv_obj_add_event_cb(obj, my_event_cb_3, LV_EVENT_ALL, NULL); /*No
→filtering, receive all events*/
```

Even the same event callback can be used on an object with different `user_data`. For example:

即使是相同的事件回调也可以用于具有不同“`user_data`”的对象。例如：

```
lv_obj_add_event_cb(obj, increment_on_click, LV_EVENT_CLICKED, &num1);
lv_obj_add_event_cb(obj, increment_on_click, LV_EVENT_CLICKED, &num2);
```

The events will be called in the order as they were added.

More objects can use the same `event callback`.

事件将按照添加的顺序调用。

更多的对象可以使用相同的事件回调。

5.7.2 Remove event(s) from an object

Events can be removed from an object with the `lv_obj_remove_event_cb(obj, event_cb)` function or `lv_obj_remove_event_dsc(obj, event_dsc)`. `event_dsc` is a pointer returned by `lv_obj_add_event_cb`.

可以使用 `lv_obj_remove_event_cb(obj, event_cb)` 函数或 `lv_obj_remove_event_dsc(obj, event_dsc)` 从对象中删除事件。`event_dsc` 是一个由 `lv_obj_add_event_cb` 返回的指针。

5.7.3 Event codes

The event codes can be grouped into these categories:

- Input device events
- Drawing events
- Other events
- Special events
- Custom events

All objects (such as Buttons/Labels/Sliders etc.) regardless their type receive the *Input device*, *Drawing* and *Other* events.

However the *Special events* are specific to a particular widget type. See the [widgets' documentation](#) to learn when they are sent,

Custom events are added by the user and therefore these are never sent by LVGL.

The following event codes exist:

事件代码可以分为以下几类：

- 输入设备事件
- 绘图事件
- 其他活动
- 特别活动
- 自定义事件

所有对象（例如按钮/标签/滑块等），无论其类型如何，都会接收 *Input device*、*Drawing* 和 *Other* 事件。

然而，特殊事件特定于特定的小部件类型。查看[widgets' 文档](#)了解何时发送，

自定义事件由用户添加，因此这些事件永远不会由 LVGL 发送。

存在以下事件代码：

Input device events (输入设备事件)

- `LV_EVENT_PRESSED` The object has been pressed
- `LV_EVENT_PRESSING` The object is being pressed (called continuously while pressing)
- `LV_EVENT_PRESS_LOST` The object is still being pressed but slid cursor/finger off of the object
- `LV_EVENT_SHORT_CLICKED` The object was pressed for a short period of time, then released it. Not called if scrolled.

- LV_EVENT_LONG_PRESSED 对象已被按下。如果对象没有滚动，则在释放时调用（无论是否长按）
- LV_EVENT_PRESSING 对象被按下（按下时连续调用）
- LV_EVENT_PRESS_LOST 对象仍被按下，但光标/手指已滑离对象
- LV_EVENT_SHORT_CLICKED 对象被按下一小段时间，然后释放它。如果滚动则不会调用。
- LV_EVENT_LONG_PRESSED 对象已按下输入设备驱动程序中指定的至少 long_press_time。如果滚动则不会调用。
- LV_EVENT_LONG_PRESSED_REPEAT 在每个 long_press_repeat_time 毫秒的 long_press_time 之后调用。如果滚动则不会调用。
- LV_EVENT_CLICKED 如果对象没有滚动，则在释放时调用（无论是否长按）
- LV_EVENT_RELEASED 在对象被释放后的每种情况下调用
- LV_EVENT_SCROLL_BEGIN 开始滚动。事件参数是 NULL 或 lv_anim_t *，如果需要，可以修改滚动动画描述符。
- LV_EVENT_SCROLL_END 滚动结束。
- LV_EVENT_SCROLL 对象被滚动
- LV_EVENT_GESTURE 检测到手势。使用 lv_indev_get_gesture_dir(lv_indev_get_act()); 获取手势
- LV_EVENT_KEY 一个密钥被发送到对象。使用 lv_indev_get_key(lv_indev_get_act()); 获得密钥
- LV_EVENT_FOCUSED 对象被聚焦
- LV_EVENT_DFOCUSSED 对象散焦
- LV_EVENT_GESTURE A gesture is detected. Get the gesture with lv_indev_get_gesture_dir(lv_indev_get_act());
- LV_EVENT_KEY A key is sent to the object. Get the key with lv_indev_get_key(lv_indev_get_act());
- LV_EVENT_LEAVE The object is defocused but still selected
- LV_EVENT_HIT_TEST Perform advanced hit-testing. Use lv_hit_test_info_t * a = lv_event_get_hit_test_info(e) and check if a->point can click the object or not. If not set a->res = false
- LV_EVENT_PRESSED 对象已被按下
- LV_EVENT_PRESSED_REPEAT 对象被按下 ms. Not called if scrolled.
- LV_EVENT_CLICKED Called on release if the object did not scroll (regardless of long press)
- LV_EVENT_RELEASED Called in every case when the object has been released
- LV_EVENT_SCROLL_BEGIN Scrolling begins. The event parameter is NULL or an lv_anim_t * with the scroll animation descriptor to modify if required.
- LV_EVENT_SCROLL_END Scrolling ends.
- LV_EVENT_SCROLL The object was scrolled

- LV_EVENT_LEAVE 对象散焦但仍被选中
- LV_EVENT_HIT_TEST 执行高级命中测试。使用 `lv_hit_test_info_t * a = lv_event_get_hit_test_info(e)` 并检查 `a->point` 是否可以点击对象。如果没有设置 `a->res = false`

Drawing events (绘图事件)

- LV_EVENT_COVER_CHECK 检查对象是否完全覆盖一个区域。事件参数是 `lv_cover_check_info_t *`。
- LV_EVENT_REFR_EXT_DRAW_SIZE 获取对象周围所需的额外绘制区域（例如用于阴影）。事件参数是 `lv_coord_t *` 来存储大小。仅用更大的值覆盖它。
- LV_EVENT_DRAW_MAIN_BEGIN 开始主绘图阶段。
- LV_EVENT_DRAW_MAIN 执行主绘图
- LV_EVENT_DRAW_MAIN_END 完成主绘制阶段
- LV_EVENT_DRAW_POST_BEGIN 开始后期绘制阶段（当所有孩子都被绘制时）
- LV_EVENT_DRAW_POST 执行后期绘制阶段（当所有孩子都被绘制时）
- LV_EVENT_DRAW_POST_END 完成后期绘制阶段（当所有孩子都被绘制时）
- LV_EVENT_DRAW_PART_BEGIN 开始绘制零件。事件参数是 `lv_obj_draw_desc_t *`。了解更多[此处](#)。
- LV_EVENT_DRAW_PART_END 完成绘制零件。事件参数是 `lv_obj_draw_desc_t *`。了解更多[此处](#)。

Other events (其他事件)

- LV_EVENT_DELETE Object is being deleted
- LV_EVENT_CHILD_CHANGED Child was removed/added
- LV_EVENT_SIZE_CHANGED Object coordinates/size have changed
- LV_EVENT_STYLE_CHANGED Object's style has changed
- LV_EVENT_BASE_DIR_CHANGED The base dir has changed
- LV_EVENT_GET_SELF_SIZE Get the internal size of a widget
- LV_EVENT_DELETE 对象正在被删除
- LV_EVENT_CHILD_CHANGED 孩子被移除/添加
- LV_EVENT_SIZE_CHANGED 对象坐标/大小已更改
- LV_EVENT_STYLE_CHANGED 对象的样式已更改
- LV_EVENT_BASE_DIR_CHANGED 基础目录已经改变
- LV_EVENT_GET_SELF_SIZE 获取小部件的内部尺寸

Special events (特殊事件)

- LV_EVENT_VALUE_CHANGED The object's value has changed (i.e. slider moved)
- LV_EVENT_INSERT A text is being inserted to the object. The event data is `char *` being inserted.
- LV_EVENT_REFRESH Notify the object to refresh something on it (for the user)
- LV_EVENT_READY A process has finished
- LV_EVENT_CANCEL A process has been canceled
- LV_EVENT_VALUE_CHANGED 对象的值已更改（即滑块移动）
- LV_EVENT_INSERT 正在向对象插入文本。事件数据是插入的 `char *`。
- LV_EVENT_REFRESH 通知对象刷新其上的某些内容（对于用户）
- LV_EVENT_READY 一个进程已经完成
- LV_EVENT_CANCEL 一个进程被取消

Custom events (自定义事件)

Any custom event codes can be registered by `uint32_t MY_EVENT_1 = lv_event_register_id();`

And can be sent to any object with `lv_event_send(obj, MY_EVENT_1, &some_data)`

任何自定义事件代码都可以通过 `uint32_t MY_EVENT_1 = lv_event_register_id();` 注册

并且可以使用 `lv_event_send(obj, MY_EVENT_1, &some_data)` 发送到任何对象

5.7.4 Sending events (发送事件)

To manually send events to an object, use `lv_event_send(obj, <EVENT_CODE> &some_data)`.

For example, this can be used to manually close a message box by simulating a button press (although there are simpler ways to do this):

要手动向对象发送事件，请使用 `lv_event_send(obj, <EVENT_CODE> &some_data)`。

例如，这可用于通过模拟按钮按下来手动关闭消息框（尽管有更简单的方法可以做到这一点）：

```
/*Simulate the press of the first button (indexes start from zero)*/
uint32_t btn_id = 0;
lv_event_send(mbox, LV_EVENT_VALUE_CHANGED, &btn_id);
```

Refresh event (刷新事件)

`LV_EVENT_REFRESH` is special event because it's designed to be used by the user to notify an object to refresh itself. Some examples:

- notify a label to refresh its text according to one or more variables (e.g. current time)
- refresh a label when the language changes
- enable a button if some conditions are met (e.g. the correct PIN is entered)
- add/remove styles to/from an object if a limit is exceeded, etc

`LV_EVENT_REFRESH` 是一个特殊事件，因为它被设计为用户使用它来通知对象刷新自身。一些例子：

- 通知标签根据一个或多个变量（例如当前时间）刷新其文本
- 当语言改变时刷新标签
- 如果满足某些条件（例如输入正确的 PIN），则启用按钮
- 如果超出限制，则向/从对象添加/删除样式等

5.7.5 Fields of `lv_event_t` (`lv_event_t` 的字段)

`lv_event_t` is the only parameter passed to event callback and it contains all the data about the event. The following values can be gotten from it:

- `lv_event_get_code(e)` get the event code
- `lv_event_get_target(e)` get the object to which the event is sent
- `lv_event_get_original_target(e)` get the object to which the event is sent originally sent (different from `lv_event_get_target` if `event bubbling` is enabled)
- `lv_event_get_user_data(e)` get the pointer passed as the last parameter of `lv_obj_add_event_cb`.
- `lv_event_get_param(e)` get the parameter passed as the last parameter of `lv_event_send`

`lv_event_t` 是传递给事件回调的唯一参数，它包含有关事件的所有数据。可以从中获得以下值：

- `lv_event_get_code(e)` 获取事件代码
- `lv_event_get_target(e)` 获取事件发送到的对象
- `lv_event_get_original_target(e)` 获取事件最初发送到的对象（与 `lv_event_get_target` 不同，如果 `event bubbling` 被启用）

- `lv_event_get_user_data(e)` 获取作为 `lv_obj_add_event_cb` 的最后一个参数传递的指针。
- `lv_event_get_param(e)` 获取作为 `lv_event_send` 的最后一个参数传递的参数

5.7.6 Event bubbling (事件冒泡)

If `lv_obj_add_flag(obj, LV_OBJ_FLAG_EVENT_BUBBLE)` is enabled all events will be sent to the object's parent too. If the parent also has `LV_OBJ_FLAG_EVENT_BUBBLE` enabled the event will be sent to its parent too, and so on.

The `target` parameter of the event is always the current target object, not the original object. To get the original target call `lv_event_get_original_target(e)` in the event handler.

如果启用了 `lv_obj_add_flag(obj, LV_OBJ_FLAG_EVENT_BUBBLE)`, 所有事件也将发送到对象的父级。如果父级也启用了 `LV_OBJ_FLAG_EVENT_BUBBLE`, 则事件也将发送到其父级, 依此类推。

事件的 `target` 参数始终是当前目标对象, 而不是原始对象。在事件处理程序中获取原始目标调用 `lv_event_get_original_target(e)`。

5.7.7 Examples

5.8 Input devices (输入设备)

An input device usually means:

- Pointer-like input device like touchpad or mouse
- Keypads like a normal keyboard or simple numeric keypad
- Encoders with left/right turn and push options
- External hardware buttons which are assigned to specific points on the screen

输入设备通常意味着:

- 类似指针的输入设备, 如触摸板或鼠标
- 像普通键盘或简单数字键盘一样的键盘
- 带左/右转和推动选项的编码器
- 分配给屏幕上特定点的外部硬件按钮

重要: Before reading further, please read the [Porting](/porting/indev) section of Input devices

重要: 在进一步阅读之前, 请阅读 Input devices 的 [Porting](/porting/indev) 部分

5.8.1 Pointers (光标)

Pointer input devices (like a mouse) can have a cursor.

有些输入设备可以有一个光标 (如鼠标)。

```
...
lv_indev_t * mouse_indev = lv_indev_drv_register(&indev_drv);

LV_IMG_DECLARE(mouse_cursor_icon); /*Declare the image file.
                                     */
lv_obj_t * cursor_obj = lv_img_create(lv_scr_act(), NULL); /*Create an image object
                                                               for the cursor */
lv_img_set_src(cursor_obj, &mouse_cursor_icon); /*Set the image source*/
lv_indev_set_cursor(mouse_indev, cursor_obj); /*Connect the image
                                              object to the driver*/
                                     */

Note that the cursor object should have lv_obj_set_click(cursor_obj, false). For images, clicking is disabled by default.

请注意，光标对象应该有 lv_obj_set_click(cursor_obj, false)。对于图像，默认情况下禁用单击。
```

5.8.2 Keypad and encoder (键盘和编码器)

You can fully control the user interface without touchpad or mouse using a keypad or encoder(s). It works similar to the TAB key on the PC to select the element in an application or a web page.

您可以使用键盘或编码器在没有触摸板或鼠标的情况下完全控制用户界面。它的工作原理类似于 PC 上的 TAB 键，用于选择应用程序或网页中的元素。

Groups (组)

The objects, you want to control with keypad or encoder, needs to be added to a *Group*. In every group, there is exactly one focused object which receives the pressed keys or the encoder actions. For example, if a *Text area* is focused and you press some letter on a keyboard, the keys will be sent and inserted into the text area. Similarly, if a *Slider* is focused 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 the keys to only one group but, a group can receive data from more than one input device too.

To create a group use `lv_group_t * g = lv_group_create()` and to add an object to the group use `lv_group_add_obj(g, obj)`.

To associate a group with an input device use `lv_indev_set_group(indev, g)`, where `indev` is the return value of `lv_indev_drv_register()`

您想用键盘或编码器控制的对象需要添加到组。在每一组中，只有一个焦点对象接收按下的键或编码器的动作。例如，如果文本区域被聚焦并且您在键盘上按下某个字母，则按键将被发送并插入到文本区域中。类似地，如果Slider 获得焦点并且您按下向左或向右箭头，则滑块的值将被更改。

您需要将输入设备与组关联。一个输入设备只能将按键发送给一组，但一组也可以从多个输入设备接收数据。

要创建一个组使用 `lv_group_t * g = lv_group_create()` 并将一个对象添加到组中使用 `lv_group_add_obj(g, obj)`。

要将组与输入设备相关联, 请使用 `lv_indev_set_group(indev, g)`, 其中 `indev` 是 `lv_indev_drv_register()` 的返回值

Keys (按键)

There are some predefined keys which have special meaning:

- **LV_KEY_NEXT** Focus on the next object
- **LV_KEY_PREV** Focus on the previous object
- **LV_KEY_ENTER** Triggers LV_EVENT_PRESSED/CLICKED/LONG_PRESSED etc. events
- **LV_KEY_UP** Increase value or move upwards
- **LV_KEY_DOWN** Decrease value or move downwards
- **LV_KEY_RIGHT** Increase value or move the the right
- **LV_KEY_LEFT** Decrease value or move the 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 a character on the left (E.g. 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*)

有一些具有特殊含义的预定义键:

- **LV_KEY_NEXT** 关注下一个对象
- **LV_KEY_PREV** 关注上一个对象
- **LV_KEY_ENTER** 触发 LV_EVENT_PRESSED/CLICKED/LONG_PRESSED 等事件
- **LV_KEY_UP** 增加值或向上移动
- **LV_KEY_DOWN** 减少值或向下移动
- **LV_KEY_RIGHT** 增加值或向右移动
- **LV_KEY_LEFT** 减少值或向左移动
- **LV_KEY_ESC** 关闭或退出 (例如关闭下拉列表)
- **LV_KEY_DEL** 删除 (例如文本区域 中右侧的字符)
- **LV_KEY_BACKSPACE** 删除左边的一个字符 (例如在文本区域)
- **LV_KEY_HOME** 转到开头/顶部 (例如在文本区域)
- **LV_KEY_END** 转到最后 (例如在文本区域)

The most important special keys are LV_KEY_NEXT/PREV, LV_KEY_ENTER and LV_KEY_UP/DOWN/LEFT/RIGHT. In your `read_cb` function, you should translate some of your keys to these special keys to navigate in the group and interact with the selected object.

Usually, it's enough to use only LV_KEY_LEFT/RIGHT because most of the objects can be fully controlled with them.

With an encoder, you should use only LV_KEY_LEFT, LV_KEY_RIGHT, and LV_KEY_ENTER.

最重要的特殊键是 LV_KEY_NEXT/PREV、LV_KEY_ENTER 和 LV_KEY_UP/DOWN/LEFT/RIGHT。在您的 `read_cb` 函数中, 您应该将一些键转换为这些特殊键, 以便 在组中导航并与所选对象进行交互。

通常，只使用“LV_KEY_LEFT/RIGHT”就足够了，因为大多数对象都可以用它们完全控制。对于编码器，您应该只使用 `LV_KEY_LEFT`、`LV_KEY_RIGHT` 和 `LV_KEY_ENTER`。

Edit and navigate mode (编辑和导航模式)

Since a keypad has plenty of keys, it's easy to navigate between the objects and edit them using the keypad. But the encoders have a limited number of "keys" and hence it is difficult to navigate using the default options. *Navigate* and *Edit* are created to avoid this problem with the encoders.

In *Navigate* mode, the encoders `LV_KEY_LEFT/RIGHT` is translated to `LV_KEY_NEXT/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/PREV` is usually used to edit the object. Depending on the object's type, a short or long press of `LV_KEY_ENTER` changes back to *Navigate* mode. Usually, an object which can not be pressed (like a *Slider*) leaves *Edit* mode on short click. But with objects where short click has meaning (e.g. *Button*), a long press is required.

由于小键盘有很多键，因此很容易在对象之间导航并使用小键盘对其进行编辑。但是编码器的“键”数量有限，因此很难使用默认选项进行导航。创建 *Navigate* 和 *Edit* 是为了避免编码器出现此问题。

在 *Navigate* 模式下，编码器 `LV_KEY_LEFT/RIGHT` 被转换为 `LV_KEY_NEXT/PREV`。因此，将通过转动编码器来选择下一个或上一个对象。

按 `LV_KEY_ENTER` 将更改为 *Edit* 模式。在 *Edit* 模式下，`LV_KEY_NEXT/PREV` 通常用于编辑对象。根据对象的类型，短按或长按 `LV_KEY_ENTER` 会变回 *Navigate* 模式。通常，无法按下的对象（如 *Slider*）会在短按时离开 *Edit* 模式。但是对于短按有意义的对象（例如 *Button*），则需要长按。

Default group (默认组)

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 the input device(s) to the default group with `lv_indev_set_group(my_indev, g);`

交互式小部件 - 例如按钮、复选框、滑块等 - 可以自动添加到默认组。只需使用 `lv_group_t * g = lv_group_create();` 创建一个组并使用 `lv_group_set_default(g);` 设置默认组

不要忘记使用 `lv_indev_set_group(my_indev, g);` 将输入设备分配到默认组。

Styling (风格样式)

If an object is focused either by clicking it via touchpad, or focused via an encoder or keypad it goes to `LV_STATE_FOCUSED`. Hence focused styles will be applied on it.

If the object goes to edit mode it goes to `LV_STATE_FOCUSED | LV_STATE_EDITED` state so these style properties will be shown.

For a more detailed description read the [Style](#) section.

如果通过触摸板单击对象或通过编码器或键盘聚焦对象，它会转到“`LV_STATE_FOCUSED`”。因此，将在其上应用重点样式。

如果对象进入编辑模式，它将进入 `LV_STATE_FOCUSED | LV_STATE_EDITED` 状态，因此将显示这些样式属性。

有关更详细的说明，请阅读 [样式](#) 部分。

5.8.3 API

Input device (输入设备)

Functions

`void lv_indev_read_timer_cb(lv_timer_t *timer)`

Called periodically to read the input devices

参数 **param** -- pointer to an input device to read

`void lv_indev_enable(lv_indev_t *indev, bool en)`

`lv_indev_t *lv_indev_get_act(void)`

Get the currently processed input device. Can be used in action functions too.

返回 pointer to the currently processed input device or NULL if no input device processing right now

`lv_indev_type_t lv_indev_get_type(const lv_indev_t *indev)`

Get the type of an input device

参数 **indev** -- pointer to an input device

返回 the type of the input device from `lv_hal_indev_type_t (LV_INDEV_TYPE_...)`

`void lv_indev_reset(lv_indev_t *indev, lv_obj_t *obj)`

Reset one or all input devices

参数

- **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_reset_long_press(lv_indev_t *indev)`

Reset the long press state of an input device

参数 **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)

参数

- **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)

参数

- **indev** -- pointer to an input device
- **group** -- point 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

参数

- **indev** -- pointer to an input device

- **group** -- point to a group

`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)

参数

- **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

参数 **indev** -- pointer to an input device

返回 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)

参数 **indev** -- pointer to an input device

返回 the last pressed key (0 on error)

`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)

参数 **indev** -- pointer to an input device

返回 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)

参数 **indev** -- pointer to an input device

返回 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)

参数

- **indev** -- pointer to an input device
- **point** -- pointer to a point to store the types.pointer.vector

`void lv_indev_wait_release(lv_indev_t *indev)`
Do nothing until the next release

参数 **indev** -- pointer to an input device

`lv_obj_t *lv_indev_get_obj_act(void)`
Gets a pointer to the currently active object in the currently processed input device.

返回 pointer to currently active object or NULL if no active object

`lv_timer_t *lv_indev_get_read_timer(lv_disp_t *indev)`
Get a pointer to the indev read timer to modify its parameters with `lv_timer_...` functions.

参数 **indev** -- pointer to an input device

返回 pointer to the indev read refresher timer. (NULL on error)

`lv_obj_t *lv_indev_search_obj (lv_obj_t *obj, lv_point_t *point)`

Search the most top, clickable object by a point

参数

- **obj** -- pointer to a start object, typically the screen
- **point** -- pointer to a point for searching the most top child

返回 pointer to the found object or NULL if there was no suitable object

Groups (组)

TypeDefs

`typedef uint8_t lv_key_t`

`typedef void (*lv_group_focus_cb_t)(struct _lv_group_t*)`

`typedef struct _lv_group_t lv_group_t`

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 `lv_cont` for that).

`typedef uint8_t lv_group_refocus_policy_t`

Enums

`enum [anonymous]`

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 [anonymous]`

Values:

enumerator `LV_GROUP_REFOCUS_POLICY_NEXT`
 enumerator `LV_GROUP_REFOCUS_POLICY_PREV`

Functions

`void _lv_group_init(void)`
Init. the group module

Remark Internal function, do not call directly.

`lv_group_t *lv_group_create(void)`
Create a new object group

返回 pointer to the new object group

`void lv_group_del(lv_group_t *group)`
Delete a group object

参数 `group` -- pointer to a group

`void lv_group_set_default(lv_group_t *group)`
Set a default group. New object are added to this group if it's enabled in their class with `add_to_def_group = true`

参数 `group` -- pointer to a group (can be NULL)

`lv_group_t *lv_group_get_default(void)`
Get the default group

返回 pointer to the default group

`void lv_group_add_obj(lv_group_t *group, struct _lv_obj_t *obj)`
Add an object to a group

参数

- `group` -- pointer to a group
- `obj` -- pointer to an object to add

`void lv_group_remove_obj(struct _lv_obj_t *obj)`
Remove an object from its group

参数 `obj` -- pointer to an object to remove

`void lv_group_remove_all_objs(lv_group_t *group)`
Remove all objects from a group

参数 `group` -- pointer to a group

`void lv_group_focus_obj(struct _lv_obj_t *obj)`
Focus on an object (defocus the current)

参数 `obj` -- pointer to an object to focus on

`void lv_group_focus_next(lv_group_t *group)`
Focus the next object in a group (defocus the current)

参数 `group` -- pointer to a group

`void lv_group_focus_prev(lv_group_t *group)`
Focus the previous object in a group (defocus the current)

参数 `group` -- pointer to a group

`void lv_group_focus_freeze(lv_group_t *group, bool en)`
Do not let to change the focus from the current object

参数

- **group** -- pointer to a group
- **en** -- true: freeze, false: release freezing (normal mode)

`lv_res_t lv_group_send_data(lv_group_t *group, uint32_t c)`

Send a control character to the focuses object of a group

参数

- **group** -- pointer to a group
- **c** -- a character (use LV_KEY_.. to navigate)

返回 result of focused object 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 object is focused

参数

- **group** -- pointer to a group
- **focus_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 item in a group is focused if the currently focused obj is deleted.

参数

- **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).

参数

- **group** -- pointer to group
- **edit** -- true: edit mode; false: navigate mode

`void lv_group_set_wrap(lv_group_t *group, bool en)`

Set whether focus next/prev will allow wrapping from first->last or last->first object.

参数

- **group** -- pointer to group
- **en** -- true: wrapping enabled; false: wrapping disabled

`struct _lv_obj_t *lv_group_get_focused(const lv_group_t *group)`

Get the focused object or NULL if there isn't one

参数 **group** -- pointer to a group

返回 pointer to the focused object

`lv_group_focus_cb_t lv_group_get_focus_cb(const lv_group_t *group)`

Get the focus callback function of a group

参数 **group** -- pointer to a group

返回 the call back function or NULL if not set

`bool lv_group_get_editing(const lv_group_t *group)`

Get the current mode (edit or navigate).

参数 **group** -- pointer to group

返回 true: edit mode; false: navigate mode

bool **lv_group_get_wrap**(*lv_group_t* *group)

Get whether focus next/prev will allow wrapping from first->last or last->first object.

参数

- **group** -- pointer to group
- **en** -- true: wrapping enabled; false: wrapping disabled

struct **_lv_group_t**

#include <lv_group.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 **lv_cont** for that).

Public Members

lv_ll_t **obj_ll**

Linked list to store the objects in the group

struct *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)

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.

5.9 Displays (显示)

重要: The basic concept of *display* in LVGL is explained in the [Porting](/porting/display) section. So before reading further, please read the [Porting](/porting/display) section first.

重要: LVGL 中 *display* 的基本概念在 [Porting](/porting/display) 部分进行了解释。因此，在进一步阅读之前，请先阅读 [移植](/porting/display) 部分。

5.9.1 Multiple display support (多显示器支持)

In LVGL, you can have multiple displays, each with their own driver and objects. The only limitation is that every display needs to be have same color depth (as defined in `LV_COLOR_DEPTH`). If the displays are different in this regard the rendered image can be converted to the correct format in the drivers `flush_cb`.

Creating more displays is easy: just initialize more display buffers and register another driver for every display. When you create the UI, use `lv_disp_set_default(disp)` to tell the library on which display to create objects.

在 LVGL 中，您可以拥有多个显示器，每个显示器都有自己的驱动程序和对象。唯一的限制是每个显示器都需要具有相同颜色深度（如 `LV_COLOR_DEPTH` 中所定义）。如果在这方面显示不同，渲染图像可以在驱动程序“`flush_cb`”中转换为正确的格式。

创建更多显示很容易：只需初始化更多显示缓冲区并为每个显示注册另一个驱动程序。创建 UI 时，使用 `lv_disp_set_default(disp)` 告诉库在哪个显示器上创建对象。

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.

为什么需要多显示器支持？这里有些例子：

- 拥有带有本地 UI 的“正常”TFT 显示，并根据需要在 VNC 上创建“虚拟”屏幕。（您需要添加您的 VNC 驱动程序）。
- 拥有大型 TFT 显示屏和小型单色显示屏。
- 在大型仪器或技术中使用一些更小更简单的显示器。
- 有两个大的 TFT 显示器：一个给顾客用，一个给店员用。

Using only one display (仅使用一个显示器)

Using more displays can be useful but in most cases it's not required. Therefore, the whole concept of multi-display is completely hidden if you register only one display. By default, the lastly created (and only) display is used.

`lv_scr_act()`, `lv_scr_load(scr)`, `lv_layer_top()`, `lv_layer_sys()`, `LV_HOR_RES` 和 `LV_VER_RES` are always applied on the most recently created (default) screen. If you pass `NULL` as `disp` parameter to display related function, usually the default display will be used. E.g. `lv_disp_trig_activity(NULL)` will trigger a user activity on the default screen. (See below in *Inactivity*).

使用更多显示器可能很有用，但在大多数情况下不是必需的。因此，如果您只注册一个显示器，则完全隐藏多显示器的概念。默认情况下，使用最后创建的（也是唯一的）显示。

`lv_scr_act()`, `lv_scr_load(scr)`, `lv_layer_top()`, `lv_layer_sys()`, `LV_HOR_RES` 和 `LV_VER_RES` 始终应用于最近创建的（默认）屏幕。如果你将 `NULL` 作为 `disp` 参数传递给显示相关函数，通常会使用默认显示。例如。`lv_disp_trig_activity(NULL)` 将在默认屏幕上触发用户活动。（请参阅下面的 *Inactivity*）。

Mirror display (镜像显示)

To mirror the image of the display to another display, you don't need to use the multi-display support. Just transfer the buffer received in `drv.flush_cb` to the other display too.

要将显示器的图像镜像到另一个显示器，您不需要使用多显示器支持。只需将在 `drv.flush_cb` 中接收到的缓冲区也传输到另一个显示器。

Split image (分割图像)

You can create a larger display from smaller ones. You can create it as below:

1. Set the resolution of the displays to the large display's resolution.
2. In `drv.flush_cb`, truncate and modify the `area` parameter for each display.
3. Send the buffer's content to each display with the truncated area.

您可以从较小的显示器创建更大的显示器。您可以按如下方式创建它：

1. 将显示器的分辨率设置为大显示器的分辨率。
2. 在 `drv.flush_cb` 中，对每个显示器截断并修改 `area` 参数。
3. 将缓冲区的内容发送到每个截断区域的显示器。

5.9.2 Screens (屏幕)

Every display has each set of `Screens` and the object on the screens.

Be sure not to confuse displays and screens:

- **Displays** are the physical hardware drawing the pixels.
- **Screens** are the high-level root objects associated with a particular display. One display can have multiple screens associated with it, but not vice versa.

Screens can be considered the highest level containers which have no parent. The screen's size is always equal to its display and size their position is (0;0). Therefore, the screens coordinates can't be changed, i.e. `lv_obj_set_pos()`, `lv_obj_set_size()` or similar functions can't be used on screens.

每个显示器都有每组 `屏幕` 和屏幕上的对象。

确保不要混淆显示和屏幕：

- **显示器**是绘制像素的物理硬件。
- **屏幕**是与特定显示器关联的高级根对象。一个显示器可以有多个与其关联的屏幕，但反之则不然。

屏幕可以被认为是没有父级的最高级别的容器。屏幕的大小总是等于它的显示和大小，它们的位置是(0;0)。因此，屏幕坐标不能改变，即 `lv_obj_set_pos()`、`lv_obj_set_size()` 或类似函数不能在屏幕上使用。

A screen can be created from any object type but the two most typical types are the `Base object` and the `Image` (to create a wallpaper).

To create a screen, use `lv_obj_t * scr = lv_<type>_create(NULL, copy)`. `copy` can be an other screen to copy it.

To load a screen, use `lv_scr_load(scr)`. To get the active screen, use `lv_scr_act()`. These functions works on the default display. If you want to specify which display to work on, use `lv_disp_get_scr_act(display)` and `lv_disp_load_scr(display, scr)`. Screen can be loaded with animations too. Read more [here](#).

Screens can be deleted with `lv_obj_del(scr)`, but ensure that you do not delete the currently loaded screen.

可以从任何对象类型创建屏幕，但最典型的两种类型是 *Base object* 和 *Image*（用于创建壁纸）。

要创建屏幕，请使用 `lv_obj_t * scr = lv_<type>_create(NULL, copy)`。`copy` 可以是另一个屏幕来复制它。

要加载屏幕，请使用 `lv_scr_load(scr)`。要获取活动屏幕，请使用 `lv_scr_act()`。这些功能适用于默认显示。如果你想指定在哪个显示器上工作，使用 `lv_disp_get_scr_act(disp)` 和 `lv_disp_load_scr(disp, scr)`。屏幕也可以加载动画。阅读更多 [此处](#)。

可以使用 `lv_obj_del(scr)` 删除屏幕，但请确保不要删除当前加载的屏幕。

Transparent screens (透明屏幕)

Usually, the opacity of the screen is `LV_OPA_COVER` to provide a solid background for its children. If it's not the case (`opacity < 100%`) the display's background color or image will be visible. See the *Display background* section for more details. If the display's background opacity is also not `LV_OPA_COVER` LVGL has no solid background to draw.

This configuration (transparent screen and display) could be used to create for example OSD menus where a video is played on a lower layer, and a menu is overlayed on an upper layer.

通常，屏幕的不透明度为“`LV_OPA_COVER`”，为其子项提供纯色背景。如果不是这种情况（不透明度 $< 100\%$ ），显示器的背景颜色或图像将可见。有关更多详细信息，请参阅显示背景部分。如果显示器的背景不透明度也不是 `LV_OPA_COVER`，则 LVGL 没有可绘制的纯色背景。

这种配置（透明屏幕和显示）可用于创建例如 OSD 菜单，其中视频在下层播放，菜单覆盖在上层。

To handle transparent displays special (slower) color mixing algorithms need to be used by LVGL so this feature needs to be enabled with `LV_COLOR_SCREEN_TRANSP` in `lv_conf.h`. As this mode operates on the Alpha channel of the pixels `LV_COLOR_DEPTH = 32` is also required. The Alpha channel of 32-bit colors will be 0 where there are no objects and 255 where there are solid objects.

In summary, to enable transparent screen and displays to create OSD menu-like UIs:

- Enable `LV_COLOR_SCREEN_TRANSP` in `lv_conf.h`
- Be sure to use `LV_COLOR_DEPTH 32`
- Set the screens opacity to `LV_OPA_TRANSP` e.g. with `lv_obj_set_style_local_bg_opa(lv_scr_act(), LV_OBJMASK_PART_MAIN, LV_STATE_DEFAULT, LV_OPA_TRANSP)`
- Set the display opacity to `LV_OPA_TRANSP` with `lv_disp_set_bg_opa(NULL, LV_OPA_TRANSP)`;

为了处理透明显示，LVGL 需要使用特殊的（较慢的）颜色混合算法，因此需要通过 `lv_conf.h` 中的 `LV_COLOR_SCREEN_TRANSP` 启用此功能。由于此模式在像素的 Alpha 通道上运行，因此还需要“`LV_COLOR_DEPTH = 32`”。32 位颜色的 Alpha 通道在没有对象时为 0，在有实体对象时为 255。

总之，要启用透明屏幕和显示以创建类似 OSD 菜单的 UI：

- 在 `lv_conf.h` 中启用 `LV_COLOR_SCREEN_TRANSP`
- 请务必使用 `LV_COLOR_DEPTH 32`
- 将 屏 幕 不 透 明 度 设 置 为 “`LV_OPA_TRANSP`” ， 例 如 使 用 `lv_obj_set_style_local_bg_opa(lv_scr_act(), LV_OBJMASK_PART_MAIN, LV_STATE_DEFAULT, LV_OPA_TRANSP)`
- 使用 `lv_disp_set_bg_opa(NULL, LV_OPA_TRANSP)`；将显示不透明度设置为 `LV_OPA_TRANSP`

5.9.3 Features of displays (显示器的特点)

Inactivity (不活动)

The user's inactivity is measured on each display. Every use of an *Input device* (if associated with the display) counts as an activity. To get time elapsed since the last activity, use `lv_disp_get_inactive_time(disp)`. If NULL is passed, the overall smallest inactivity time will be returned from all displays (**not the default display**).

You can manually trigger an activity using `lv_disp_trig_activity(disp)`. If `disp` is NULL, the default screen will be used (**and not all displays**).

在每个显示器上测量用户的不活动。每次使用输入设备（如果与显示器相关联）都算作一次活动。要获取自上次活动以来经过的时间，请使用 `lv_disp_get_inactive_time(disp)`。如果传递 NULL，则将从所有显示（不是默认显示）返回总体最小不活动时间。

您可以使用 `lv_disp_trig_activity(disp)` 手动触发活动。如果 `disp` 为 NULL，将使用默认屏幕（**并非所有显示**）。

Background (背景)

Every display has background color, a background image and background opacity properties. They become visible when the current screen is transparent or not positioned to cover the whole display.

Background color is a simple color to fill the display. It can be adjusted with `lv_disp_set_bg_color(disp, color)`;

Background image is a path to a file or a pointer to an `lv_img_dsc_t` variable (converted image) to be used as wallpaper. It can be set with `lv_disp_set_bg_color(disp, &my_img)`; If the background image is set (not NULL) the background won't be filled with `bg_color`.

The opacity of the background color or image can be adjusted with `lv_disp_set_bg_opa(disp, opa)`.

The `disp` parameter of these functions can be NULL to refer it to the default display.

每个显示器都有背景颜色、背景图像和背景不透明度属性。当当前屏幕透明或未定位以覆盖整个显示器时，它们变得可见。

背景色是一种简单的颜色来填充显示。可以通过 `lv_disp_set_bg_color(disp, color)` 进行调整；

背景图像是文件的路径或指向用作墙纸的“`lv_img_dsc_t`”变量（转换后的图像）的指针。可以用 `lv_disp_set_bg_color(disp, &my_img)` 设置；如果设置了背景图像（不是“NULL”），背景将不会被“`bg_color`”填充。

背景颜色或图像的不透明度可以通过 `lv_disp_set_bg_opa(disp, opa)` 进行调整。

这些函数的 `disp` 参数可以是 NULL 以引用默认显示。

5.9.4 API

Enums

enum `lv_scr_load_anim_t`
Values:

enumerator `LV_SCR_LOAD_ANIM_NONE`

enumerator `LV_SCR_LOAD_ANIM_OVER_LEFT`

```

enumerator LV_SCR_LOAD_ANIM_OVER_RIGHT
enumerator LV_SCR_LOAD_ANIM_OVER_TOP
enumerator LV_SCR_LOAD_ANIM_OVER_BOTTOM
enumerator LV_SCR_LOAD_ANIM_MOVE_LEFT
enumerator LV_SCR_LOAD_ANIM_MOVE_RIGHT
enumerator LV_SCR_LOAD_ANIM_MOVE_TOP
enumerator LV_SCR_LOAD_ANIM_MOVE_BOTTOM
enumerator LV_SCR_LOAD_ANIM_FADE_ON

```

Functions

`lv_obj_t *lv_disp_get_scr_act(lv_disp_t *disp)`

Return with a pointer to the active screen

参数 `disp` -- pointer to display which active screen should be get. (NULL to use the default screen)

返回 pointer to the active screen object (loaded by 'lv_scr_load()')

`lv_obj_t *lv_disp_get_scr_prev(lv_disp_t *disp)`

Return with a pointer to the previous screen. Only used during screen transitions.

参数 `disp` -- pointer to display which previous screen should be get. (NULL to use the default screen)

返回 pointer to the previous screen object or NULL if not used now

`void lv_disp_load_scr(lv_obj_t *scr)`

Make a screen active

参数 `scr` -- pointer to a screen

`lv_obj_t *lv_disp_get_layer_top(lv_disp_t *disp)`

Return with the top layer. (Same on every screen and it is above the normal screen layer)

参数 `disp` -- pointer to display which top layer should be get. (NULL to use the default screen)

返回 pointer to the top layer object (transparent screen sized lv_obj)

`lv_obj_t *lv_disp_get_layer_sys(lv_disp_t *disp)`

Return with the sys. layer. (Same on every screen and it is above the normal screen and the top layer)

参数 `disp` -- pointer to display which sys. layer should be get. (NULL to use the default screen)

返回 pointer to the sys layer object (transparent screen sized lv_obj)

`void lv_disp_set_theme(lv_disp_t *disp, lv_theme_t *th)`

Get the theme of a display

参数 `disp` -- pointer to a display

返回 the display's theme (can be NULL)

`lv_theme_t *lv_disp_get_theme(lv_disp_t *disp)`

Get the theme of a display

参数 `disp` -- pointer to a display

返回 the display's theme (can be NULL)

void lv_disp_set_bg_color(*lv_disp_t* *disp, *lv_color_t* color)
Set the background color of a display

参数

- **disp** -- pointer to a display
- **color** -- color of the background

void lv_disp_set_bg_image(*lv_disp_t* *disp, const void *img_src)
Set the background image of a display

参数

- **disp** -- pointer to a display
- **img_src** -- path to file or pointer to an *lv_img_dsc_t* variable

void lv_disp_set_bg_opa(*lv_disp_t* *disp, *lv_opa_t* opa)
Opacity of the background

参数

- **disp** -- pointer to a display
- **opa** -- opacity (0..255)

void lv_scr_load_anim(*lv_obj_t* *scr, *lv_scr_load_anim_t* anim_type, uint32_t time, uint32_t delay, bool auto_del)

Switch screen with animation

参数

- **scr** -- pointer to the new screen to load
- **anim_type** -- type of the animation from *lv_scr_load_anim_t*. E.g. *LV_SCR_LOAD_ANIM_MOVE_LEFT*
- **time** -- time of the animation
- **delay** -- delay before the transition
- **auto_del** -- true: automatically delete the old screen

uint32_t lv_disp_get_inactive_time(const *lv_disp_t* *disp)
Get elapsed time since last user activity on a display (e.g. click)

参数 **disp** -- pointer to an display (NULL to get the overall smallest inactivity)

返回 elapsed ticks (milliseconds) since the last activity

void lv_disp_trig_activity(*lv_disp_t* *disp)
Manually trigger an activity on a display

参数 **disp** -- pointer to an display (NULL to use the default display)

void lv_disp_clean_dcache(*lv_disp_t* *disp)
Clean any CPU cache that is related to the display.

参数 **disp** -- pointer to an display (NULL to use the default display)

***lv_timer_t* * _lv_disp_get_refr_timer(*lv_disp_t* *disp)**
Get a pointer to the screen refresher timer to modify its parameters with *lv_timer_...* functions.

参数 **disp** -- pointer to a display

返回 pointer to the display refresher timer. (NULL on error)

static inline `lv_obj_t` ***lv_scr_act**(void)
Get the active screen of the default display

返回 pointer to the active screen

static inline `lv_obj_t` ***lv_layer_top**(void)
Get the top layer of the default display

返回 pointer to the top layer

static inline `lv_obj_t` ***lv_layer_sys**(void)
Get the active screen of the default display

返回 pointer to the sys layer

static inline void **lv_scr_load**(`lv_obj_t` *scr)

static inline lv_coord_t **lv_dpx**(lv_coord_t n)

Scale the given number of pixels (a distance or size) relative to a 160 DPI display considering the DPI of the default display. It ensures that e.g. `lv_dpx(100)` will have the same physical size regardless to the DPI of the display.

参数 n -- the number of pixels to scale

返回 n × current_dpi/160

static inline lv_coord_t **lv_disp_dpx**(const `lv_disp_t` *disp, lv_coord_t n)

Scale the given number of pixels (a distance or size) relative to a 160 DPI display considering the DPI of the given display. It ensures that e.g. `lv_dpx(100)` will have the same physical size regardless to the DPI of the display.

参数

- **obj** -- an display whose dpi should be considered
- **n** -- the number of pixels to scale

返回 n × current_dpi/160

5.10 Colors (颜色)

The color module handles all color-related functions like changing color depth, creating colors from hex code, converting between color depths, mixing colors, etc.

`lv_color_t` is used to store a color, its fields are set according to `LV_COLOR_DEPTH` in `lv_conf.h`. (See below)

You may set `LV_COLOR_16_SWAP` in `lv_conf.h` to swap the bytes of *RGB565* colors. You may need this to send the 16-bit colors via a byte-oriented interface like SPI. As 16-bit numbers are stored in Little Endian format (lower byte on the lower address), the interface will send the lower byte first. However, displays usually need the higher byte first. A mismatch in the byte order will result in highly distorted colors.

颜色模块处理所有与颜色相关的功能，如更改颜色深度、从十六进制代码创建颜色、颜色深度之间的转换、混合颜色等。

`lv_color_t` 用于存储颜色，其字段根据 `lv_conf.h` 中的 `LV_COLOR_DEPTH` 设置。(见下文)

你可以在 `lv_conf.h` 中设置 `LV_COLOR_16_SWAP` 来交换 *RGB565* 颜色的字节。您可能需要它来通过面向字节的接口（如 SPI）发送 16 位颜色。由于 16 位数字以 Little Endian 格式存储（低位字节在低位地址），因此接口将首先发送低位字节。但是，显示器通常首先需要较高的字节。字节顺序不匹配将导致颜色高度失真。

5.10.1 Creating colors (创造色彩)

RGB (三原色)

Create colors from Red, Green and Blue channel values

从红色、绿色和蓝色通道值创建颜色

```
//All channels are 0-255
lv_color_t c = lv_color_make(red, green, blue);

//From hex code 0x000000..0xFFFF 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 (色调饱和值-Hue Saturation Value)

Create colors from Hue, Saturation and Value values

根据色相、饱和度和值创建颜色

```
//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 (调色板)

LVGL includes material design's palette. In this all color have a main 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

LVGL 包括材料设计的调色板。在此所有颜色都有一个主要的以及四个较深的变体和五个较浅的变体。

颜色名称如下：

- 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.

要获得主要颜色，请使用 `lv_color_t c = lv_palette_main(LV_PALETTE_...)`。

对于调色板颜色的较浅变体，请使用 `lv_color_t c = lv_palette_lighten(LV_PALETTE_..., v)`。`v` 可以是 1..5。

对于调色板颜色的较暗变体，请使用 `lv_color_t c = lv_palette_darken(LV_PALETTE_..., v)`。v 可以是 1..4。

Modify and mix colors (修改和混合颜色)

The following functions can modify a color:

以下函数可以修改颜色：

```
// 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: black
lv_color_t c = lv_color_change_lightness(lv_color_t c, lv_opa_t lvl);

// Mix 2 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 (内置颜色)

`lv_color_white()` 和 `lv_color_black()` 返回 `0xFFFFFFFF` 和 `0x00000000` 分别。

`lv_color_white()` 和 `lv_color_black()` 分别返回 `0xFFFFFFFF` 和 `0x00000000`。

5.10.2 Opacity (不透明度)

To describe opacity the `lv_opa_t` type is created as a wrapper to `uint8_t`. Some defines are also introduced:

- `LV_OPA_TRANSP` Value: 0, means the opacity makes the color completely transparent
- `LV_OPA_10` Value: 25, means the color covers only a little
- `LV_OPA_20 ... OPA_80` come logically
- `LV_OPA_90` Value: 229, means the color near completely covers
- `LV_OPA_COVER` Value: 255, means the color completely covers

You can also use the `LV_OPA_*` defines in `lv_color_mix()` as a *ratio*.

5.10.3 Color types (颜色类型)

The following variable types are defined by the color module:

- `lv_color1_t` Monochrome color. Also has R, G, B fields for compatibility but they are always the same value (1 byte)
- `lv_color8_t` A structure to store R (3 bit),G (3 bit),B (2 bit) components for 8-bit colors (1 byte)
- `lv_color16_t` A structure to store R (5 bit),G (6 bit),B (5 bit) components for 16-bit colors (2 byte)
- `lv_color32_t` A structure to store R (8 bit),G (8 bit), B (8 bit) components for 24-bit colors (4 byte)
- `lv_color_t` Equal to `lv_color1/8/16/24_t` depending on current color depth setting

- `lv_color_int_t` `uint8_t`, `uint16_t` or `uint32_t` depending on color depth setting. Used to build color arrays from plain numbers.
- `lv_opa_t` A simple `uint8_t` type to describe opacity.

The `lv_color_t`, `lv_color1_t`, `lv_color8_t`, `lv_color16_t` and `lv_color32_t` types have four fields:

- `ch.red` red channel
- `ch.green` green channel
- `ch.blue` blue channel
- `full`* red + green + blue as one number

You can set the current color depth in `lv_conf.h`, by setting the `LV_COLOR_DEPTH` define to 1 (monochrome), 8, 16 or 32.

Convert color (颜色转换)

You can convert a color from the current color depth to another. The converter functions return with a number, so you have to use the `full` field:

```
lv_color_t c;
c.red = 0x38;
c.green = 0x70;
c.blue = 0xCC;

lv_color1_t c1;
c1.full = lv_color_to1(c); /*Return 1 for light colors, 0 for dark colors*/

lv_color8_t c8;
c8.full = lv_color_to8(c); /*Give a 8 bit number with the converted color*/

lv_color16_t c16;
c16.full = lv_color_to16(c); /*Give a 16 bit number with the converted color*/

lv_color32_t c24;
c24.full = lv_color_to32(c); /*Give a 32 bit number with the converted color*/
```

5.10.4 API

Typedefs

```
typedef lv_color_t (*lv_color_filter_cb_t)(const struct lv_color_filter_dsc_t*, lv_color_t, lv_opa_t)
typedef struct lv_color_filter_dsc_t lv_color_filter_dsc_t
```

Enums

enum [**anonymous**]

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_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

```
typedef LV_CONCAT3 (uint, LV_COLOR_SIZE, _t) lv_color_int_t
```

```
typedef LV_CONCAT3 (lv_color, LV_COLOR_DEPTH, _t) lv_color_t
```

```
static inline uint8_t lv_color_to1(lv_color_t color)
```

```
static inline uint8_t lv_color_to8(lv_color_t color)
```

```
static inline uint16_t lv_color_to16(lv_color_t color)
```

```
static inline uint32_t lv_color_to32(lv_color_t color)
```

```
static inline uint8_t lv_color_brightness(lv_color_t color)
```

Get the brightness of a color

参数 **color** -- a color

返回 the brightness [0..255]

```
static inline lv_color_t lv_color_make(uint8_t r, uint8_t g, uint8_t b)
```

```
static inline lv_color_t lv_color_hex(uint32_t c)
```

```
static inline lv_color_t lv_color_hex3(uint32_t c)
```

```
static inline void lv_color_filter_dsc_init(lv_color_filter_dsc_t *dsc, lv_color_filter_cb_t cb)
```

```
lv_color_t lv_color_lighten(lv_color_t c, lv_opa_t lvl)
```

```
lv_color_t lv_color_darken(lv_color_t c, lv_opa_t lvl)
```

```
lv_color_t lv_color_change_lightness(lv_color_t c, lv_opa_t lvl)
```

```
lv_color_t lv_color_hsv_to_rgb(uint16_t h, uint8_t s, uint8_t v)
```

Convert a HSV color to RGB

参数

- **h** -- hue [0..359]

- **s** -- saturation [0..100]
- **v** -- value [0..100]

返回 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

参数

- **r8** -- 8-bit red
- **g8** -- 8-bit green
- **b8** -- 8-bit blue

返回 the given RGB color in HSV

lv_color_hsv_t **lv_color_to_hsv**(lv_color_t color)

Convert a color to HSV

参数 **color** -- color

返回 the given color in HSV

static inline lv_color_t **lv_color_chroma_key**(void)

Just a wrapper around LV_COLOR_CHROMA_KEY because it might be more convenient to use a function in some cases

返回 LV_COLOR_CHROMA_KEY

lv_color_t **lv_palette_main**(*lv_palette_t* p)

static inline lv_color_t **lv_color_white**(void)

static inline lv_color_t **lv_color_black**(void)

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)

union **lv_color1_t**

Public Members

uint8_t **full**

uint8_t **blue**

uint8_t **green**

uint8_t **red**

union *lv_color1_t*::[anonymous] **ch**

union **lv_color8_t**

Public Members

```

uint8_t blue
uint8_t green
uint8_t red
struct lv_color8_t::[anonymous] ch
uint8_t full
union lv_color16_t

```

Public Members

```

uint16_t blue
uint16_t green
uint16_t red
uint16_t green_h
uint16_t green_l
struct lv_color16_t::[anonymous] ch
uint16_t full
union lv_color32_t

```

Public Members

```

uint8_t blue
uint8_t green
uint8_t red
uint8_t alpha
struct lv_color32_t::[anonymous] ch
uint32_t full
struct lv_color_hsv_t

```

Public Members

```

uint16_t h
uint8_t s
uint8_t v
struct _lv_color_filter_dsc_t

```

Public Members

```
lv_color_filter_cb_t filter_cb
void *user_data
```

5.11 Fonts (字体)

In LVGL fonts are collections of bitmaps and other information required to render the images of the letters (glyph). A font is stored in a `lv_font_t` variable and can be set in a style's `text_font` field. For example:

在 LVGL 中，字体是渲染字母（字形）图像所需的位图和其他信息的集合。字体存储在 `lv_font_t` 变量中，可以在样式的 `text_font` 字段中设置。例如：

```
lv_style_set_text_font(&my_style, LV_STATE_DEFAULT, &lv_font_montserrat_28); /*Set a
➥larger font*/
```

The fonts have a **bpp** (**bits per pixel**) property. It shows how many bits are used to describe a pixel in the font. The value stored for a pixel determines the pixel's opacity. This way, with higher *bpp*, the edges of the letter can be smoother. The possible *bpp* values are 1, 2, 4 and 8 (higher value means better quality).

The *bpp* also affects the required memory size to store the font. For example, *bpp* = 4 makes the font nearly 4 times larger compared to *bpp* = 1.

字体具有 **bpp** (**每像素位数**) 属性。它显示了使用多少位来描述字体中的像素。为像素存储的值决定了像素的不透明度。这样，使用更高的 *bpp*，字母的边缘可以更平滑。可能的 *bpp* 值为 1、2、4 和 8（值越高表示质量越好）。

bpp 还会影响存储字体所需的内存大小。例如，*bpp* = 4 使字体比 *bpp* = 1 大近 4 倍。

5.11.1 Unicode support (支持 Unicode 编码)

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

LVGL 支持 **UTF-8** 编码的 Unicode 字符。您的编辑器需要配置为将您的代码/文本保存为 UTF-8（通常这是默认值），并确保在 `lv_conf.h` 中将 `LV_TXT_ENC` 设置为 `LV_TXT_ENC_UTF8`。（这是默认值）

要测试它，请参考这个用法：

```
lv_obj_t * label1 = lv_label_create(lv_scr_act(), NULL);
lv_label_set_text(label1, LV_SYMBOL_OK);
```

If all works well, a ✓ character should be displayed.

如果一切正常，应显示 ✓ 字符。

5.11.2 Built-in fonts (内置字体)

There are several built-in fonts in different sizes, which can be enabled in `lv_conf.h` by `LV_FONT_...` defines.

有几种不同大小的内置字体，可以通过 `LV_FONT_...` 定义在 `lv_conf.h` 中启用。

Normal fonts (普通字体)

Containing all the 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

包含所有 ASCII 字符、度数符号 (U+00B0)、子弹符号 (U+2022) 和内置符号（见下文）。

- `LV_FONT_MONTserrat_12` 12 像素字体
- `LV_FONT_MONTserrat_14` 14 像素字体
- `LV_FONT_MONTserrat_16` 16 像素字体
- `LV_FONT_MONTserrat_18` 18 像素字体
- `LV_FONT_MONTserrat_20` 20 像素字体
- `LV_FONT_MONTserrat_22` 22 像素字体
- `LV_FONT_MONTserrat_24` 24 像素字体
- `LV_FONT_MONTserrat_26` 26 像素字体
- `LV_FONT_MONTserrat_28` 28 像素字体

- LV_FONT_MONTserrat_30 30 像素字体
- LV_FONT_MONTserrat_32 32 像素字体
- LV_FONT_MONTserrat_34 34 像素字体
- LV_FONT_MONTserrat_36 36 像素字体
- LV_FONT_MONTserrat_38 38 像素字体
- LV_FONT_MONTserrat_40 40 像素字体
- LV_FONT_MONTserrat_42 42 像素字体
- LV_FONT_MONTserrat_44 44 像素字体
- LV_FONT_MONTserrat_46 46 像素字体
- LV_FONT_MONTserrat_48 48 像素字体

Special fonts (特殊字体)

- LV_FONT_MONTserrat_12_SUBPX Same as normal 12 px font but with *subpixel rendering*
- LV_FONT_MONTserrat_28_COMPRESSED Same as normal 28 px font but *compressed font* 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_SIMSUN_16_CJK16 px font with normal range + 1000 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 shown above.

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_FONT_MONTserrat_12_SUBPX 与普通 12 px 字体相同，但具有子像素渲染
- LV_FONT_MONTserrat_28_COMPRESSED 与普通 28 px 字体相同，但压缩字体为 3 bpp
- LV_FONT_DEJAVU_16_PERSIAN_HEBREW 16 px 字体，正常范围 + 希伯来语、阿拉伯语、波斯语字母及其所有形式
- LV_FONT_SIMSUN_16_CJK16 px 字体，正常范围 + 1000 个最常见的 CJK 部首
- LV_FONT_UNSCII_8 8 px 像素完美字体，只有 ASCII 字符
- LV_FONT_UNSCII_16 16 px 像素完美字体，只有 ASCII 字符

内置字体是全局变量，其名称类似于 16 像素高字体的 “`lv_font_montserrat_16`”。要在样式中使用它们，只需添加一个指向字体变量的指针，如上所示。

$bpp = 4$ 的内置字体包含 ASCII 字符并使用 `Montserrat` 字体。

除了 ASCII 范围外，以下符号还添加到 `FontAwesome` 字体的内置字体中。

	LV_SYMBOL_AUDIO
	LV_SYMBOL_VIDEO
	LV_SYMBOL_LIST
	LV_SYMBOL_OK
	LV_SYMBOL_CLOSE
	LV_SYMBOL_POWER
	LV_SYMBOL_SETTINGS
	LV_SYMBOL_TRASH
	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_EDIT
	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_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_BACKSPACE
	LV_SYMBOL_SD_CARD
	LV_SYMBOL_NEW_LINE

The symbols can be used as:

这些符号可以这样使用：

```
lv_label_set_text(my_label, LV_SYMBOL_OK);
```

Or with together with strings:

或与字符串一起使用：

```
lv_label_set_text(my_label, LV_SYMBOL_OK "Apply");
```

Or more symbols together:

一个或多个符号组合在一起:

```
lv_label_set_text(my_label, LV_SYMBOL_OK LV_SYMBOL_WIFI LV_SYMBOL_PLAY);
```

5.11.3 Special features (特殊功能)

Bidirectional support (双向支持)

Most of the languages use Left-to-Right (LTR for short) writing direction, however some languages (such as Hebrew, Persian or Arabic) uses Right-to-Left (RTL for short) direction.

LVGL not only supports RTL texts but supports mixed (a.k.a. bidirectional, BiDi) text rendering too. Some examples:

大多数语言使用从左到右（简称 LTR）书写方向，但是一些语言（例如希伯来语、波斯语或阿拉伯语）使用从右到左（简称 RTL）方向。

LVGL 不仅支持 RTL 文本，还支持混合（又名双向，BiDi）文本渲染。一些例子：

The names of these states in Arabic
are الكويت and البحرين, مصر respectively.

The title is مفتاح معايير الريب! in Arabic.

BiDi support is enabled by `LV_USE_BIDI` in `lv_conf.h`

All texts have a base direction (LTR or RTL) which determines some rendering rules and the default alignment of the text (Left or Right). However, in LVGL, base direction is applied not only for labels. It's a general property which can be set for every object. If unset then it will be inherited from the parent. So it's enough to set the base direction of the screen and every object will inherit it.

The default base direction of screen can be set by `LV_BIDI_BASE_DIR_DEF` in `lv_conf.h` and other objects inherit the base direction from their parent.

BiDi 支持由 `lv_conf.h` 中的 `LV_USE_BIDI` 启用

所有文本都有一个基本方向（LTR 或 RTL），它决定了一些渲染规则和文本的默认对齐方式（左或右）。然而，在 LVGL 中，基本方向不仅适用于标签。这是一个可以为每个对象设置的通用属性。如果未设置，则它将从父级继承。所以设置屏幕的基本方向就足够了，每个对象都会继承它。

屏幕的默认基本方向可以通过 `lv_conf.h` 中的 `LV_BIDI_BASE_DIR_DEF` 设置，其他对象从其父对象继承基本方向。

To set an object's base direction use `lv_obj_set_base_dir(obj, base_dir)`. The possible base direction are:

- `LV_BIDI_DIR_LTR`: Left to Right base direction
- `LV_BIDI_DIR_RTL`: Right to Left base direction
- `LV_BIDI_DIR_AUTO`: Auto detect base direction
- `LV_BIDI_DIR_INHERIT`: Inherit the base direction from the parent (default for non-screen objects)

This list summarizes the effect of RTL base direction on objects:

- Create objects by default on the right
- `lv_tabview`: displays tabs from right to left
- `lv_checkbox`: Show the box on the right
- `lv_btndmatrix`: Show buttons from right to left
- `lv_list`: Show the icon on the right
- `lv_dropdown`: Align the options to the right
- The texts in `lv_table`, `lv_btndmatrix`, `lv_keyboard`, `lv_tabview`, `lv_dropdown`, `lv_roller` are "BiDi processed" to be displayed correctly

要设置对象的基本方向, 请使用 `lv_obj_set_base_dir(obj, base_dir)`。可能的基本方向是:

- `LV_BIDI_DIR_LTR`: 从左到右的基本方向
- `LV_BIDI_DIR_RTL`: 从右到左的基本方向
- `LV_BIDI_DIR_AUTO`: 自动检测基本方向
- `LV_BIDI_DIR_INHERIT`: 从父级继承基本方向 (非屏幕对象的默认值)

此列表总结了 RTL 基本方向对对象的影响:

- 默认在右侧创建对象
- `lv_tabview`: 从右到左显示标签
- `lv_checkbox`: 显示右侧的框
- `lv_btndmatrix`: 从右到左显示按钮
- `lv_list`: 在右侧显示图标
- `lv_dropdown`: 将选项向右对齐
- `lv_table`, `lv_btndmatrix`, `lv_keyboard`, `lv_tabview`, `lv_dropdown`, `lv_roller` 中的文本是 "BiDi 处理" 以正确显示

Arabic and Persian support(阿拉伯语和波斯语支持)

There are some special rules to display Arabic and Persian characters: the *form* of the character depends on their position in the text. A different form of the same letter needs to be used if it isolated, start, middle or end position. Besides these some conjunction rules also should be taken into account.

LVGL supports to apply these rules if `LV_USE_ARABIC_PERSIAN_CHARS` is enabled.

显示阿拉伯语和波斯语字符有一些特殊规则: 字符的形式取决于它们在文本中的位置。如果同一个字母是孤立的、开始的、中间的或结束的位置, 则需要使用不同形式的相同字母。除了这些, 还应该考虑一些连词规则。

如果启用了 "`LV_USE_ARABIC_PERSIAN_CHARS`", LVGL 支持应用这些规则。

However, there some limitations:

- Only displaying texts is supported (e.g. on labels), text inputs (e.g. text area) don't support this feature.
- Static text (i.e. const) is not processed. E.g. texts set by `lv_label_set_text()` will be "Arabic processed" but `lv_label_set_text_static()` won't.
- Text get functions (e.g. `lv_label_get_text()`) will return the processed text.

但是，有一些限制：

- 仅支持显示文本（例如在标签上），文本输入（例如文本区域）不支持此功能。
- 不处理静态文本（即 const）。例如。`lv_label_set_text()` 设置的文本将是“阿拉伯语处理”，但 `lv_label_set_text_static()` 不会。
- 文本获取函数（例如 `lv_label_get_text()`）将返回处理后的文本。

Subpixel rendering (亚像素渲染)

Subpixel rendering allows for tripling the horizontal resolution by rendering on Red, Green and Blue channel instead of pixel level. This takes advantage of the position of physical color channels of each pixel, resulting in higher quality letter anti-aliasing. Learn more [here](#).

For subpixel rendering the fonts need to be generated with special settings:

- In the online converter tick the **Subpixel** box
- In the command line tool use `--lcd` flag. Note that the generated font needs about 3 times more memory.

Subpixel rendering works only if the color channels of the pixels have a horizontal layout. That is the R, G, B channels are next each other and not above each other. The order of color channels also needs to match with the library settings. By default LVGL assumes RGB order, however this can be swapped by setting `LV_SUBPX_BGR 1` in `lv_conf.h`.

亚像素渲染通过在红色、绿色和蓝色通道（而不是像素级）上渲染，允许将水平分辨率提高三倍。这将利用每个像素的物理颜色通道的位置，从而实现更高质量的字母消除混叠。了解更多[此处](#)。

对于亚像素渲染，需要使用特殊设置生成字体：-在在线转换器中，勾选“亚像素”框 -在命令行工具中，使用“`--lcd`”标志。请注意，生成的字体需要大约3倍的内存。

仅当像素的颜色通道具有水平布局时，子像素渲染才起作用。也就是说，R、G、B 通道彼此相邻，而不是彼此上方。颜色通道的顺序也需要与库设置相匹配。默认情况下，LVGL 采用“RGB”顺序，但这可以通过在 `LV_conf.h` 中设置“`LV_SUBPX_BGR 1`”进行交换。

Compress fonts (压缩字体)

The bitmaps of the fonts can be compressed by

- ticking the **Compressed** check box in the online converter
- not passing `--no-compress` flag to the offline converter (compression is applied by default)

The compression is more effective with larger fonts and higher bpp. However, it's about 30% slower to render the compressed fonts. Therefore it's recommended to compress only the largest fonts of user interface, because

- they need the most memory
- they can be compressed better
- and probably they are used less frequently than the medium sized fonts, so the performance cost is smaller.

字体的位图可以通过以下方式压缩

- 勾选在线转换器中的“压缩”复选框
- 不将 `--no-compress` 标志传递给离线转换器（默认情况下应用压缩）

对于更大的字体和更高的 bpp，压缩更有效。但是，渲染压缩字体的速度要慢 30% 左右。因此建议只压缩用户界面的最大字体，因为

- 他们需要最多的内存

- 它们可以被更好地压缩
- 可能它们的使用频率低于中等字体，因此性能成本更小。

5.11.4 Add new font (添加新字体)

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 (Roboto font and symbols) but in 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 `lv_font_conv` to be installed)

To declare the font in a file, use `LV_FONT_DECLARE(my_font_name)`.

To make the fonts globally available (like the builtin fonts), add them to `LV_FONT_CUSTOM_DECLARE` in `lv_conf.h`.

有几种方法可以将新字体添加到您的项目中：

1. 最简单的方法是使用【在线字体转换器】(<https://lvgl.io/tools/fontconverter>)。只需设置参数，单击 *Convert* 按钮，将字体复制到您的项目并使用它。**请务必仔细阅读该网站上提供的步骤，否则转换时会出错。**
2. 使用【离线字体转换器】(https://github.com/lvgl/lv_font_conv)。(需要安装 Node.js)
3. 如果您想创建类似内置字体 (Roboto 字体和符号) 但大小和/或范围不同的内容，您可以使用 `lvgl/scripts/built_in_font` 文件夹中的 `built_in_font_gen.py` 脚本。(这需要安装 Python 和 `lv_font_conv`)

要在文件中声明字体，请使用 `LV_FONT_DECLARE(my_font_name)`。

要使字体全局可用 (如内置字体)，请将它们添加到 `lv_conf.h` 中的 `LV_FONT_CUSTOM_DECLARE`。

5.11.5 Add new symbols (添加新符号)

The built-in symbols are created from the [FontAwesome](#) font.

1. Search symbol on <https://fontawesome.com>. For example the USB symbol. Copy its Unicode ID which is `0xf287` in this case.
2. Open the [Online font converter](#). Add `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 it 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 `LV_FONT_DECLARE(my_font_name);`

Using the symbol

内置符号是从 [FontAwesome](#) 字体创建的。

1. 在<https://fontawesome.com>上搜索符号。例如 USB 符号。复制它的 Unicode ID，在本例中为“`0xf287`”。
2. 打开【在线字体转换器】(<https://lvgl.io/tools/fontconverter>)。添加添加 `FontAwesome.woff..`

3. 设置 Name、Size、BPP 等参数。您将使用此名称在代码中声明和使用字体。
4. 将符号的 Unicode ID 添加到范围字段中。例如，USB 符号的 `0xf287`。更多的符号可以用，来枚举。
5. 转换字体并将其复制到您的项目中。确保编译字体的.c 文件。
6. 使用 `extern lv_font_t my_font_name;` 或简单的 `LV_FONT_DECLARE(my_font_name);` 声明字体。

使用符号

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` 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 `style.text.font` properties. To use the symbol you may need to change it. Eg `style.text.font = my_font_name`

1. 将 Unicode 值转换为 UTF8，例如在[本站](#)。对于 `0xf287`, *Hex UTF-8 字节*是 `EF 8A 87`。
2. 从 UTF8 值创建一个 `define`: `#define MY_USB_SYMBOL "\xEF\x8A\x87"`
3. 创建标签并设置文本。例如。`lv_label_set_text (标签, MY_USB_SYMBOL)`

注意 - `lv_label_set_text(label, MY_USB_SYMBOL)` 在 `style.text.font` 属性中定义的字体中搜索此符号。要使用该符号，您可能需要对其进行更改。例如 `style.text.font = my_font_name`

5.11.6 Load font at run-time (在运行时加载字体)

`lv_font_load` can be used to load a font from a file. The font to load needs to have a special binary format. (Not TTF or WOFF). Use `lv_font_conv` with `--format bin` option to generate an LVGL compatible font file.

Note that to load a font [LVGL's filesystem](#) needs to be enabled and a driver needs to be added.

Example

`lv_font_load` 可用于从文件加载字体。要加载的字体需要具有特殊的二进制格式。(不是 TTF 或 WOFF)。使用 `lv_font_conv` 和 `--format bin` 选项来生成 LVGL 兼容字体文件。

请注意，要加载字体[LVGL 的文件系统](#)需要启用，并且需要添加驱动程序。

例子

```
lv_font_t * my_font;
my_font = lv_font_load(X/path/to/my_font.bin);

/*Use the font*/

/*Free the font if not required anymore*/
lv_font_free(my_font);
```

5.11.7 Add a new font engine (添加新的字体引擎)

LVGL's font interface is designed to be very flexible. But even so you don't need to use LVGL's internal font engine: you can add your own. For example, 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.

A ready to use FreeType can be found in [lv_freetype](#) repository.

To do this a custom `lv_font_t` variable needs to be created:

LVGL 的字体界面设计得非常灵活。但即便如此，您也不需要使用 LVGL 的内部字体引擎：您可以添加自己的字体引擎。例如，使用 [FreeType](#) 从 TTF 字体实时渲染字形或使用外部闪存存储字体的位图并在库需要时读取它们。

可以在 [lv_freetype](#) 存储库中找到准备使用的 FreeType。

为此，需要创建一个自定义的 `lv_font_t` 变量：

```
/*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 [px]*/
    dsc_out->ofs_y = 3;          /*Y offset of the bitmap measured from the as line*/
    dsc_out->bpp = 2;            /*Bits per pixel: 1/2/4/8*/

    return true;                  /*true: glyph found; false: glyph was not found*/
}
```

(下页继续)

(续上页)

```

/* 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*/
}

```

5.12 Images (图象)

An image can be a file or variable which stores the bitmap itself and some metadata.

图像可以是存储位图本身和一些元数据的文件或变量。

5.12.1 Store images (存储图像)

You can store images in two places

- as a variable in the internal memory (RAM or ROM)
- as a file

您可以将图像存储在两个地方

- 作为内部存储器（RAM 或 ROM）中的变量
- 作为文件

Variables (变量)

The images stored internally in a variable are composed mainly of an **lv_img_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.

内部存储在变量中的图像主要由具有以下字段的 **lv_img_dsc_t** 结构组成：

- 标题
- *cf* 颜色格式。见下面
- *w* 像素宽度 (≤ 2048)
- *h* 以像素为单位的高度 (≤ 2048)
- 始终为零 3 位，需要始终为零
- 保留保留供将来使用
- **data** 指向存储图像本身的数组的指针
- **data_size** **data** 的长度（以字节为单位）

这些通常作为 C 文件存储在项目中。它们像任何其他常量数据一样链接到生成的可执行文件中。

Files (文件)

To deal with files you need to add a *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 to RAM before being drawn. As a result, they are not as resource-friendly as variable images. However, they are easier to replace without needing to recompile the main program.

要处理文件，您需要将 *Drive* 添加到 LVGL。简而言之，*Drive* 是在 LVGL 中注册的用于进行文件操作的函数 (*open*、*read*、*close* 等) 的集合。您可以向标准文件系统 (SD 卡上的 FAT32) 添加接口，或者创建简单的文件系统以从 SPI 闪存读取数据。在任何情况下，*Drive* 都只是读取和/或将数据写入内存的抽象。请参阅[文件系统](#)部分了解更多信息。

存储为文件的图像不会链接到生成的可执行文件中，并且必须在绘制之前读取到 RAM。因此，它们不像可变图像那样资源友好。但是，它们更容易替换而无需重新编译主程序。

5.12.2 Color formats (颜色格式)

Various built-in color formats are supported:

- **LV_IMG_CF_TRUE_COLOR** Simply stores the RGB colors (in whatever color depth LVGL is configured for).
- **LV_IMG_CF_TRUE_COLOR_ALPHA** Like **LV_IMG_CF_TRUE_COLOR** but it also adds an alpha (transparency) byte for every pixel.
- **LV_IMG_CF_TRUE_COLOR_CHROMA_KEYED** Like **LV_IMG_CF_TRUE_COLOR** but if a pixel has **LV_COLOR_TRANSP** (set in *lv_conf.h*) color the pixel will be transparent.
- **LV_IMG_CF_INDEXED_1/2/4/8BIT** Uses a palette with 2, 4, 16 or 256 colors and stores each pixel in 1, 2, 4 or 8 bits.
- **LV_IMG_CF_ALPHA_1/2/4/8BIT** Only stores the Alpha value on 1, 2, 4 or 8 bits. The pixels take the color of **style.image.color** 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 but you'd like to be able to change it).

The bytes of the **LV_IMG_CF_TRUE_COLOR** images are stored in the following order.

支持各种内置颜色格式：

- **LV_IMG_CF_TRUE_COLOR** 简单地存储 RGB 颜色（以 LVGL 配置的任何颜色深度）。

- **LV_IMG_CF_TRUE_COLOR_ALPHA** 类似于 **LV_IMG_CF_TRUE_COLOR**, 但它还为每个像素添加了一个 alpha (透明度) 字节。
- **LV_IMG_CF_TRUE_COLOR_CHROMA_KEYED** 类似于 **LV_IMG_CF_TRUE_COLOR**, 但如果像素具有 **LV_COLOR_TRANSP** (在 *lv_conf.h* 中设置) 颜色, 则像素将是透明的。
- **LV_IMG_CF_INDEXED_1/2/4/8BIT** 使用具有 2、4、16 或 256 种颜色的调色板, 并以 1、2、4 或 8 位存储每个像素。
- **LV_IMG_CF_ALPHA_1/2/4/8BIT** 仅以 1、2、4 或 8 位存储 Alpha 值。像素采用 `style.image.color` 的颜色和设置的不透明度。源图像必须是 alpha 通道。这非常适用于类似于字体的位图 (其中整个图像是一种颜色, 但您希望能够更改它)。

LV_IMG_CF_TRUE_COLOR 图像的字节按以下顺序存储。

For 32-bit color depth:

- Byte 0: Blue
- Byte 1: Green
- Byte 2: Red
- Byte 3: Alpha

For 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_IMG_CF_TRUE_COLOR_ALPHA**)

For 8-bit color depth:

- Byte 0: Red 3 bit, Green 3 bit, Blue 2 bit
- Byte 2: Alpha byte (only with **LV_IMG_CF_TRUE_COLOR_ALPHA**)

对于 32 位色深:

- 字节 0: 蓝色
- 字节 1: 绿色
- 字节 2: 红色
- 字节 3: 阿尔法

对于 16 位色深:

- 字节 0: 绿色 3 低位, 蓝色 5 位
- 字节 1: 红色 5 位, 绿色 3 高位
- 字节 2: Alpha 字节 (仅适用于 **LV_IMG_CF_TRUE_COLOR_ALPHA**)

对于 8 位色深:

- 字节 0: 红色 3 位, 绿色 3 位, 蓝色 2 位
- 字节 2: Alpha 字节 (仅适用于 **LV_IMG_CF_TRUE_COLOR_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_IMG_CF_RAW** Indicates a basic raw image (e.g. a PNG or JPG image).
- **LV_IMG_CF_RAW_ALPHA** Indicates that the image has alpha and an alpha byte is added for every pixel.

- **LV_IMG_CF_RAW_CHROME_KEYED** Indicates that the image is chroma-keyed as described in **LV_IMG_CF_TRUE_COLOR_CHROMA_KEYED** above.

您可以以 *Raw* 格式存储图像，以表明它没有使用其中一种内置颜色格式进行编码，并且需要使用外部 [图像解码器](#) 来解码图像。

- **LV_IMG_CF_RAW** 表示基本的原始图像（例如 PNG 或 JPG 图像）。
- **LV_IMG_CF_RAW_ALPHA** 表示图像具有 alpha 并且为每个像素添加一个 alpha 字节。
- **LV_IMG_CF_RAW_CHROME_KEYED** 表示图像是色度键控的，如上面“**LV_IMG_CF_TRUE_COLOR_CHROMA_KEYED**”中所述。

5.12.3 Add and use images (添加和使用图像)

You can add images to LVGL in two ways:

- using the online converter
- manually create images

您可以通过两种方式将图像添加到 LVGL：

- 使用在线转换器
- 手动创建图像

Online converter (在线转换器)

The online Image converter is available here: <https://lvgl.io/tools/imageconverter>

Adding an image to LVGL via 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.

在线图像转换器可在此处获得：<https://lvgl.io/tools/imageconverter>

通过在线转换器将图像添加到 LVGL 很容易。

1. 您需要先选择 *BMP*、*PNG* 或 *JPG* 图像。
2. 为图像指定一个将在 LVGL 中使用的名称。
3. 选择[颜色格式](#)。
4. 选择您想要的图像类型。选择二进制文件将生成一个 `.bin` 文件，该文件必须单独存储并使用[文件支持](#)读取。选择一个变量将生成一个可以链接到您的项目的标准 C 文件。
5. 点击转换按钮。转换完成后，您的浏览器将自动下载生成的文件。

In the converter C arrays (variables), the 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 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

在转换器 C 数组（变量）中，所有颜色深度（1、8、16 或 32）的位图都包含在 C 文件中，但实际上只有与 *lv_conf.h* 中的 **LV_COLOR_DEPTH** 匹配的颜色深度才会链接到生成的可执行文件中。

对于二进制文件，您需要指定所需的颜色格式：

- RGB332 8 位色深
- RGB565 16 位色深
- RGB565 交换 16 位色深（交换两个字节）
- RGB888 用于 32 位色深

Manually create an image (手动创建图像)

If you are generating an image at run-time, you can craft an image variable to display it using LVGL. For example:

如果您在运行时生成图像，您可以制作一个图像变量以使用 LVGL 显示它。例如：

```
uint8_t my_img_data[] = {0x00, 0x01, 0x02, ...};

static lv_img_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_IMG_CF_TRUE_COLOR,           /*Set the color format*/
    .data = my_img_data,
};
```

If the color format is **LV_IMG_CF_TRUE_COLOR_ALPHA** you can set **data_size** like **80 * 60 * LV_IMG_PX_SIZE_ALPHA_BYTE**.

Another (possibly simpler) option to create and display an image at run-time is to use the [Canvas](#) object.

如果颜色格式是 **LV_IMG_CF_TRUE_COLOR_ALPHA**，你可以将 **data_size** 设置为 **80 * 60 * LV_IMG_PX_SIZE_ALPHA_BYTE**。

在运行时创建和显示图像的另一个（可能更简单）选项是使用[Canvas](#) 对象。

Use images (使用图片)

The simplest way to use an image in LVGL is to display it with an [lv_img](#) object:

在 LVGL 中使用图像的最简单方法是使用 [lv_img](#) 对象显示它：

```
lv_obj_t * icon = lv_img_create(lv_scr_act(), NULL);

/*From variable*/
lv_img_set_src(icon, &my_icon_dsc);

/*From file*/
lv_img_set_src(icon, "S:my_icon.bin");
```

If the image was converted with the online converter, you should use `LV_IMG_DECLARE(my_icon_dsc)` to declare the image in the file where you want to use it.

如果图像是使用在线转换器转换的，则应使用 `LV_IMG_DECLARE(my_icon_dsc)` 在要使用的文件中声明图像。

5.12.4 Image decoder (图像解码器)

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.

The image decoder consists of 4 callbacks:

- **info** get some basic info about the image (width, height and color format).
- **open** open the image: either store the decoded image or set it to `NULL` to indicate the image can be read line-by-line.
- **read** if *open* didn't fully open the image this function should give some decoded data (max 1 line) from a given position.
- **close** close the 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 `LV_IMG_CF_TRUE_COLOR_...`, `LV_IMG_INDEXED_...` and `LV_IMG_ALPHA_...` formats (essentially, all non-`RAW` formats) are understood by the built-in decoder.

正如您在[颜色格式](#)部分中看到的，LVGL 支持多种内置图像格式。在许多情况下，这些将是您所需要的。但是，LVGL 不直接支持 PNG 或 JPG 等通用图像格式。

要处理非内置图像格式，您需要使用外部库并通过图像解码器接口将它们附加到 LVGL。

图像解码器由 4 个回调组成：

- **info** 获取有关图像的一些基本信息（宽度、高度和颜色格式）。
- **open** 打开图像：要么存储解码后的图像，要么将其设置为 `NULL` 以指示可以逐行读取图像。
- **read** 如果 *open* 没有完全打开图像，这个函数应该从给定的位置给出一些解码数据（最多 1 行）。
- **close** 关闭打开的图片，释放分配的资源。

您可以添加任意数量的图像解码器。当需要绘制图像时，库将尝试所有注册的图像解码器，直到找到可以打开图像的解码器，即知道该格式的解码器。

`LV_IMG_CF_TRUE_COLOR_...`、`LV_IMG_INDEXED_...` 和 `LV_IMG_ALPHA_...` 格式（基本上，所有非 `RAW` 格式）被内置解码器理解。

Custom image formats (自定义图像格式)

The easiest way to create a custom image is to use the online image converter and set `Raw`, `Raw with alpha` or `Raw with chroma-keyed` format. It will just take every byte of the binary file you uploaded and write it as the 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_IMG_CF_RAW`, `LV_IMG_CF_RAW_ALPHA` or `LV_IMG_CF_RAW_CHROME_KEYED` accordingly. You should choose the correct format according to your needs: fully opaque image, use alpha channel or use chroma keying.

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 [`#color-formats`](Color formats) section.

创建自定义图像的最简单方法是使用在线图像转换器并设置“Raw”、“Raw with alpha”或“Raw with chroma-keyed”格式。它只会获取您上传的二进制文件的每个字节并将其写入图像“位图”。然后，您需要附加一个图像解码器，该解码器将解析该位图并生成真实的、可渲染的位图。

`header.cf` 将相应地为 `LV_IMG_CF_RAW`、`LV_IMG_CF_RAW_ALPHA` 或 `LV_IMG_CF_RAW_CHROME_KEYED`。您应该根据需要选择正确的格式：完全不透明的图像、使用 alpha 通道或使用色度键控。

解码后，`raw` 格式被库视为真彩色。换句话说，图像解码器必须根据 [`#color-formats`] (颜色格式) 部分中描述的格式将 `Raw` 图像解码为 *True color*。

If you want to create a custom image, you should use `LV_IMG_CF_USER_ENCODED_0..7` color formats. However, the library can draw the images only in *True color* format (or `Raw` but finally it's supposed to be in *True color* format). The `LV_IMG_CF_USER_ENCODED_...` formats are not known by the library and therefore they should be decoded to one of the known formats from [`#color-formats`](Color formats) section. It's possible to decode the image to a non-true color format first (for example: `LV_IMG_INDEXED_4BITS`) and then call the built-in decoder functions to convert it to *True color*.

With *User encoded* formats, the color format in the open function (`dsc->header.cf`) should be changed according to the new format.

如果要创建自定义图像，则应使用 `LV_IMG_CF_USER_ENCODED_0..7` 颜色格式。但是，该库只能以 *True color* 格式（或 `Raw` 但最终它应该以 *True color* 格式）绘制图像。

lvgl 库不知道 `LV_IMG_CF_USER_ENCODED_...` 格式，因此应该将它们解码为 [`#color-formats`] (颜色格式) 部分中的已知格式之一。可以先将图像解码为非真彩色格式（例如：`LV_IMG_INDEXED_4BITS`），然后调用内置解码器函数将其转换为真彩色。

使用用户编码格式，打开函数 (`dsc->header.cf`) 中的颜色格式应根据新格式进行更改。

Register an image decoder (注册图像解码器)

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:

这是让 LVGL 处理 PNG 图像的示例。

首先，您需要创建一个新的图像解码器并设置一些功能来打开/关闭 PNG 文件。它应该是这样的：

```
/*Create a new decoder and register functions */
lv_img_decoder_t * dec = lv_img_decoder_create();
lv_img_decoder_set_info_cb(dec, decoder_info);
lv_img_decoder_set_open_cb(dec, decoder_open);
lv_img_decoder_set_close_cb(dec, decoder_close);
```

(下页继续)

(续上页)

```

/**
 * 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 store the info here
 * @return LV_RES_OK: no error; LV_RES_INV: can't get the info
 */
static lv_res_t decoder_info(lv_img_decoder_t * decoder, const void * src, lv_img_
    ↪header_t * header)
{
    /*Check whether the type `src` is known by the decoder*/
    if(is_png(src) == false) return LV_RES_INV;

    /* Read the PNG header and find `width` and `height` */
    ...

    header->cf = LV_IMG_CF_RAW_ALPHA;
    header->w = width;
    header->h = height;
}

/**
 * Open a PNG image and return the decided image
 * @param decoder pointer to the decoder where this function belongs
 * @param dsc pointer to a descriptor which describes this decoding session
 * @return LV_RES_OK: no error; LV_RES_INV: can't get the info
 */
static lv_res_t decoder_open(lv_img_decoder_t * decoder, lv_img_decoder_dsc_t * dsc)
{
    /*Check whether the type `src` is known by the decoder*/
    if(is_png(src) == false) return LV_RES_INV;

    /*Decode and store the image. If `dsc->img_data` is `NULL`, the `read_line` ↪
     ↪function will be called to get the image data line-by-line*/
    dsc->img_data = my_png_decoder(src);

    /*Change the color format if required. For PNG usually 'Raw' is fine*/
    dsc->header.cf = LV_IMG_CF_...

    /*Call a built in decoder function if required. It's not required if `my_png_ ↪
     ↪decoder` opened the image in true color format.*/
    lv_res_t res = lv_img_decoder_builtin_open(decoder, dsc);

    return res;
}

/**
 * Decode `len` pixels starting from the given `x`, `y` coordinates and store them in ↪
     ↪`buf`.
 * Required only if the "open" function can't open the whole decoded pixel array. ↪
     ↪(dsc->img_data == NULL)
 * @param decoder pointer to the decoder the function associated with
 * @param dsc pointer to decoder descriptor

```

(下页继续)

(续上页)

```

* @param x start x coordinate
* @param y start y coordinate
* @param len number of pixels to decode
* @param buf a buffer to store the decoded pixels
* @return LV_RES_OK: ok; LV_RES_INV: failed
*/
lv_res_t decoder_builtin_read_line(lv_img_decoder_t * decoder, lv_img_decoder_dsc_t *
    ↪* dsc, lv_coord_t x,
                                    lv_coord_t y, lv_coord_t len, uint8_
    ↪t * buf)
{
    /*With PNG it's usually not required*/

    /*Copy `len` pixels from `x` and `y` coordinates in True color format to `buf` */
}

/**
 * Free the allocated resources
 * @param decoder pointer to the decoder where this function belongs
 * @param dsc pointer to a descriptor which describes this decoding session
 */
static void decoder_close(lv_img_decoder_t * decoder, lv_img_decoder_dsc_t * dsc)
{
    /*Free all allocated data*/

    /*Call the built-in close function if the built-in open/read_line was used*/
    lv_img_decoder_builtin_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_RES_INV`. However, if you can open the image, a pointer to the decoded *True color* image should be set in `dsc->img_data`. If the format is known but you don't want to decode the entire image (e.g. no memory for it) set `dsc->img_data = NULL` to call `read_line` to get the pixels.
- In `decoder_close` you should free all the allocated resources.
- `decoder_read` is optional. Decoding the whole image requires extra memory and some computational overhead. However, if can decode one line of the image without decoding the whole image, you can save memory and time. To indicate that the *line read* function should be used, set `dsc->img_data = NULL` in the open function.

所以总结一下：

- 在 `decoder_info` 中，你应该收集一些关于图像的基本信息并将其存储在 `header` 中。
- 在 `decoder_open` 中，你应该尝试打开 `dsc->src` 指向的图像源。它的类型已经在 `dsc->src_type == LV_IMG_SRC_FILE/VARIABLE` 中。如果解码器不支持此格式/类型，则返回 “`LV_RES_INV`”。但是，如果您可以打开图像，则应在 `dsc->img_data` 中设置指向解码的真彩色图像的指针。如果格式已知但您不想解码整个图像（例如没有内存），请设置 `dsc->img_data = NULL` 以调用 `read_line` 来获取像素。
- 在 `decoder_close` 中，你应该释放所有分配的资源。

- `decoder_read` 是可选的。解码整个图像需要额外的内存和一些计算开销。但是，如果可以解码一行图像而不解码整个图像，则可以节省内存和时间。表示 `line read` 函数应该是我们

Manually use an image decoder (手动使用图像解码器)

LVGL will use the registered image decoder automatically if you try and draw a raw image (i.e. using the `lv_img` object) but you can use them manually too. Create a `lv_img_decoder_dsc_t` variable to describe the decoding session and call `lv_img_decoder_open()`.

如果您尝试绘制原始图像（即使使用 `lv_img` 对象），LVGL 将自动使用注册的图像解码器，但您也可以手动使用它们。创建一个 `lv_img_decoder_dsc_t` 变量来描述解码会话并调用 `lv_img_decoder_open()`。

```
lv_res_t res;
lv_img_decoder_dsc_t dsc;
res = lv_img_decoder_open(&dsc, &my_img_dsc, LV_COLOR_WHITE);

if(res == LV_RES_OK) {
    /*Do something with `dsc->img_data`*/
    lv_img_decoder_close(&dsc);
}
```

5.12.5 Image caching (图片缓存)

Sometimes it takes a lot of time to open an image. Continuously decoding a PNG image or loading images from a slow external memory would be inefficient and detrimental to the user experience.

Therefore, LVGL caches a given number of images. Caching means some images will be left open, hence LVGL can quickly access them from `dsc->img_data` 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. If you have a deeply embedded target which decodes small images from a relatively fast storage medium, image caching may not be worth it.

有时打开图像需要很多时间。连续解码 PNG 图像或从缓慢的外部存储器加载图像将是低效的并且不利于用户体验。

因此，LVGL 缓存给定数量的图像。缓存意味着一些图像将保持打开状态，因此 LVGL 可以从 `dsc->img_data` 快速访问它们，而无需再次解码它们。

当然，缓存图像是资源密集型的，因为它使用更多的 RAM（用于存储解码的图像）。LVGL 尝试尽可能地优化流程（见下文），但您仍需要评估这是否对您的平台有益。如果您有一个深度嵌入的目标，可以从相对较快的存储介质中解码小图像，则图像缓存可能不值得。

Cache size (缓存大小)

The number of cache entries can be defined in `LV_IMG_CACHE_DEF_SIZE` in `lv_conf.h`. The default value is 1 so only the most recently used image will be left open.

The size of the cache can be changed at run-time with `lv_img_cache_set_size(entry_num)`.

缓存条目的数量可以在 `lv_conf.h` 中的 `LV_IMG_CACHE_DEF_SIZE` 中定义。默认值为 1，因此只有最近使用的图像将保持打开状态。

缓存的大小可以在运行时通过 `lv_img_cache_set_size(entry_num)` 改变。

Value of images (图片的价值)

When you use more images than cache entries, LVGL can't cache all of 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 *time to open* value in the decoder open function in `dsc->time_to_open = time_ms` to give a higher or lower value. (Leave it unchanged to let LVGL set it.)

Every cache entry has a "*life*" value. Every time an image opening happens through the cache, the *life* value of all entries is decreased to make them older. When a cached image is used, its *life* value is increased by the *time to open* value to make it more alive.

If there is no more space in the cache, the entry with the smallest life value will be closed.

当您使用的图像多于缓存条目时，LVGL 无法缓存所有图像。相反，库将关闭缓存的图像之一（以释放空间）。

为了决定关闭哪个图像，LVGL 使用它之前对打开图像所花费的时间进行的测量。保存打开速度较慢的图像的缓存条目被认为更有价值，并尽可能长时间地保存在缓存中。

如果您想要或需要覆盖 LVGL 的测量，您可以在 `dsc->time_to_open = time_ms` 中的解码器打开函数中手动设置 *time to open* 值，以给出更高或更低的值。（保持不变，让 LVGL 设置它。）

每个缓存条目都有一个 "*life*" 值。每次通过缓存打开图像时，所有条目的 *life* 值都会减少以使其更旧。当使用缓存图像时，它的 *life* 值会增加 *time to open* 值以使其更加活跃。

如果缓存中没有更多空间，则生命值最小的条目将被关闭。

Memory usage (内存使用情况)

Note that the cached image might continuously consume memory. For example, if 3 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.

请注意，缓存的图像可能会持续消耗内存。例如，如果缓存了 3 个 PNG 图片，它们将在打开时消耗内存。

因此，用户有责任确保有足够的 RAM 来同时缓存最大的图像。

Clean the cache (清理缓存)

Let's say you have loaded a PNG image into a `lv_img_dsc_t my_png` variable and use it in an `lv_img` object. 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.

To do this, use `lv_img_cache_invalidate_src(&my_png)`. If `NULL` is passed as a parameter, the whole cache will be cleaned.

假设您已将 PNG 图像加载到 `lv_img_dsc_t my_png` 变量中，并在 `lv_img` 对象中使用它。如果图像已经缓存，然后您更改了底层 PNG 文件，则需要通知 LVGL 再次缓存图像。否则，没有简单的方法可以检测到底层文件发生了变化，而 LVGL 仍会绘制旧图像。

为此，请使用 `lv_img_cache_invalidate_src(&my_png)`。如果将 `NULL` 作为参数传递，则整个缓存将被清除。

5.12.6 API

Image buffer (图像缓冲区)

Typedefs

typedef uint8_t **lv_img_cf_t**

Enums

enum [**anonymous**]

Values:

enumerator **LV_IMG_CF_UNKNOWN**

enumerator **LV_IMG_CF_RAW**

Contains the file as it is. Needs custom decoder function

enumerator **LV_IMG_CF_RAW_ALPHA**

Contains the file as it is. The image has alpha. Needs custom decoder function

enumerator **LV_IMG_CF_RAW_CHROMA_KEYED**

Contains the file as it is. The image is chroma keyed. Needs custom decoder function

enumerator **LV_IMG_CF_TRUE_COLOR**

Color format and depth should match with LV_COLOR settings

enumerator **LV_IMG_CF_TRUE_COLOR_ALPHA**

Same as **LV_IMG_CF_TRUE_COLOR** but every pixel has an alpha byte

enumerator **LV_IMG_CF_TRUE_COLOR_CHROMA_KEYED**

Same as **LV_IMG_CF_TRUE_COLOR** but LV_COLOR_TRANSP pixels will be transparent

enumerator **LV_IMG_CF_INDEXED_1BIT**

Can have 2 different colors in a palette (always chroma keyed)

enumerator **LV_IMG_CF_INDEXED_2BIT**

Can have 4 different colors in a palette (always chroma keyed)

enumerator **LV_IMG_CF_INDEXED_4BIT**

Can have 16 different colors in a palette (always chroma keyed)

enumerator **LV_IMG_CF_INDEXED_8BIT**

Can have 256 different colors in a palette (always chroma keyed)

enumerator **LV_IMG_CF_ALPHA_1BIT**

Can have one color and it can be drawn or not

enumerator **LV_IMG_CF_ALPHA_2BIT**

Can have one color but 4 different alpha value

enumerator LV_IMG_CF_ALPHA_4BIT

Can have one color but 16 different alpha value

enumerator LV_IMG_CF_ALPHA_8BIT

Can have one color but 256 different alpha value

enumerator LV_IMG_CF_RESERVED_15

Reserved for further use.

enumerator LV_IMG_CF_RESERVED_16

Reserved for further use.

enumerator LV_IMG_CF_RESERVED_17

Reserved for further use.

enumerator LV_IMG_CF_RESERVED_18

Reserved for further use.

enumerator LV_IMG_CF_RESERVED_19

Reserved for further use.

enumerator LV_IMG_CF_RESERVED_20

Reserved for further use.

enumerator LV_IMG_CF_RESERVED_21

Reserved for further use.

enumerator LV_IMG_CF_RESERVED_22

Reserved for further use.

enumerator LV_IMG_CF_RESERVED_23

Reserved for further use.

enumerator LV_IMG_CF_USER_ENCODED_0

User holder encoding format.

enumerator LV_IMG_CF_USER_ENCODED_1

User holder encoding format.

enumerator LV_IMG_CF_USER_ENCODED_2

User holder encoding format.

enumerator LV_IMG_CF_USER_ENCODED_3

User holder encoding format.

enumerator LV_IMG_CF_USER_ENCODED_4

User holder encoding format.

enumerator LV_IMG_CF_USER_ENCODED_5

User holder encoding format.

enumerator **LV_IMG_CF_USER_ENCODED_6**

User holder encoding format.

enumerator **LV_IMG_CF_USER_ENCODED_7**

User holder encoding format.

Functions

*lv_img_dsc_t *lv_img_buf_alloc(lv_coord_t w, lv_coord_t h, lv_img_cf_t cf)*

Allocate an image buffer in RAM

参数

- **w** -- width of image
- **h** -- height of image
- **cf** -- a color format (LV_IMG_CF_...)

返回 an allocated image, or NULL on failure

*lv_color_t lv_img_buf_get_px_color(lv_img_dsc_t *dsc, lv_coord_t x, lv_coord_t y, lv_color_t color)*

Get the color of an image's pixel

参数

- **dsc** -- an image descriptor
- **x** -- x coordinate of the point to get
- **y** -- x coordinate of the point to get
- **color** -- the color of the image. In case of LV_IMG_CF_ALPHA_1/2/4/8 this color is used. Not used in other cases.
- **safe** -- true: check out of bounds

返回 color of the point

*lv_opa_t lv_img_buf_get_px_alpha(lv_img_dsc_t *dsc, lv_coord_t x, lv_coord_t y)*

Get the alpha value of an image's pixel

参数

- **dsc** -- pointer to an image descriptor
- **x** -- x coordinate of the point to set
- **y** -- x coordinate of the point to set
- **safe** -- true: check out of bounds

返回 alpha value of the point

*void lv_img_buf_set_px_color(lv_img_dsc_t *dsc, lv_coord_t x, lv_coord_t y, lv_color_t c)*

Set the color of a pixel of an image. The alpha channel won't be affected.

参数

- **dsc** -- pointer to an image descriptor
- **x** -- x coordinate of the point to set
- **y** -- x coordinate of the point to set

- **c** -- color of the point
- **safe** -- true: check out of bounds

void lv_img_buf_set_px_alpha(*lv_img_dsc_t* *dsc, lv_coord_t x, lv_coord_t y, lv_opa_t opa)
Set the alpha value of a pixel of an image. The color won't be affected

参数

- **dsc** -- pointer to an image descriptor
- **x** -- x coordinate of the point to set
- **y** -- y coordinate of the point to set
- **opa** -- the desired opacity
- **safe** -- true: check out of bounds

void lv_img_buf_set_palette(*lv_img_dsc_t* *dsc, uint8_t id, lv_color_t c)
Set the palette color of an indexed image. Valid only for LV_IMG_CF_INDEXED1/2/4/8

参数

- **dsc** -- pointer to an image descriptor
- **id** -- the palette color to set:
 - for LV_IMG_CF_INDEXED1: 0..1
 - for LV_IMG_CF_INDEXED2: 0..3
 - for LV_IMG_CF_INDEXED4: 0..15
 - for LV_IMG_CF_INDEXED8: 0..255
- **c** -- the color to set

void lv_img_buf_free(*lv_img_dsc_t* *dsc)
Free an allocated image buffer

参数 **dsc** -- image buffer to free

uint32_t lv_img_buf_get_img_size(lv_coord_t w, lv_coord_t h, *lv_img_cf_t* cf)
Get the memory consumption of a raw bitmap, given color format and dimensions.

参数

- **w** -- width
- **h** -- height
- **cf** -- color format

返回 size in bytes

void _lv_img_buf_transform_init(*lv_img_transform_dsc_t* *dsc)
Initialize a descriptor to rotate an image

参数 **dsc** -- pointer to an *lv_img_transform_dsc_t* variable whose **cfg** field is initialized

bool _lv_img_buf_transform_anti_alias(*lv_img_transform_dsc_t* *dsc)
Continue transformation by taking the neighbors into account

参数 **dsc** -- pointer to the transformation descriptor

`bool _lv_img_buf_transform(lv_img_transform_dsc_t *dsc, lv_coord_t x, lv_coord_t y)`
Get which color and opa would come to a pixel if it were rotated

注解: the result is written back to `dsc->res_color` and `dsc->res_opa`

参数

- **dsc** -- a descriptor initialized by `lv_img_buf_rotate_init`
- **x** -- the coordinate which color and opa should be get
- **y** -- the coordinate which color and opa should be get

返回 true: there is valid pixel on these x/y coordinates; false: the rotated pixel was out of the image

`void _lv_img_buf_get_transformed_area(lv_area_t *res, lv_coord_t w, lv_coord_t h, int16_t angle, uint16_t zoom, const lv_point_t *pivot)`

Get the area of a rectangle if its rotated and scaled

参数

- **res** -- store the coordinates here
- **w** -- width of the rectangle to transform
- **h** -- height of the rectangle to transform
- **angle** -- angle of rotation
- **zoom** -- zoom, (256 no zoom)
- **pivot** -- x,y pivot coordinates of rotation

struct **lv_img_header_t**

#include <lv_img_buf.h> The first 8 bit is very important to distinguish the different source types. For more info see `lv_img_get_src_type()` in lv_img.c On big endian systems the order is reversed so cf and always_zero must be at the end of the struct.

Public Members

```
uint32_t h
uint32_t w
uint32_t reserved
uint32_t always_zero
uint32_t cf
```

struct **lv_img_header_t**

#include <lv_img_buf.h> The first 8 bit is very important to distinguish the different source types. For more info see `lv_img_get_src_type()` in lv_img.c On big endian systems the order is reversed so cf and always_zero must be at the end of the struct.

Public Members

```
uint32_t h
uint32_t w
uint32_t reserved
uint32_t always_zero
uint32_t cf

struct lv_img_dsc_t
#include <lv_img_buf.h> Image header it is compatible with the result from image converter utility
```

Public Members

lv_img_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

```
struct lv_img_transform_dsc_t
```

Public Members

```
const void *src
lv_coord_t src_w
lv_coord_t src_h
lv_coord_t pivot_x
lv_coord_t pivot_y
int16_t angle
uint16_t zoom
lv_color_t color
lv_img_cf_t cf
bool antialias

struct lv_img_transform_dsc_t::[anonymous] cfg
lv_opa_t opa
struct lv_img_transform_dsc_t::[anonymous] res
lv_img_dsc_t img_dsc
int32_t pivot_x_256
```

```

int32_t pivot_y_256
int32_t sinma
int32_t cosma
uint8_t chroma_keyed
uint8_t has_alpha
uint8_t native_color
uint32_t zoom_inv
lv_coord_t xs
lv_coord_t ys
lv_coord_t xs_int
lv_coord_t ys_int
uint32_t pxi
uint8_t px_size
struct lv_img_transform_dsc_t::[anonymous] tmp

```

5.13 File system (文件系统)

LVGL has a 'File system' abstraction module that enables you to attach any type of file system. The file system is identified by a drive letter. For example, if the SD card is associated with the letter 'S', a file can be reached like "S:/path/to/file.txt".

LVGL 有一个“文件系统”抽象模块，使您能够附加任何类型的文件系统。文件系统由驱动器号标识。例如，如果 SD 卡与字母“S”相关联，则可以访问类似“S:/path/to/file.txt”的文件。

5.13.1 Ready to use drivers (准备使用驱动程序)

The [lv_fs_if](#) repository contains ready to use drivers using POSIX, standard C and [FATFS](#) API. See it's [README](#) for the details.

[lv_fs_if](#) 存储库包含使用 POSIX、标准 C 和 [FATFS](#) 的即用型驱动程序) API。有关详细信息，请参阅 [README](#)。

5.13.2 Add a driver (添加驱动程序)

Registering a driver (注册驱动)

To add a driver, `lv_fs_drv_t` needs to be initialized like below. `lv_fs_drv_t` needs to be static, global or dynamically allocated and not a local variable.

要添加驱动程序，`lv_fs_drv_t` 需要像下面这样初始化。`lv_fs_drv_t` 需要是静态的、全局的或动态分配的，而不是局部变量。

```

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*/
use */                         /*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)*/
use */                         /*Callback to tell the cursor position */
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*/
use */                         /*Callback to read a directory's 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.

任何回调都可以为“NULL”以指示不支持该操作。

Implementing the callbacks (实现回调)

Open callback (打开回调)

The prototype of **open_cb** looks like this:

open_cb 的原型如下所示：

```
void * (*open_cb)(lv_fs_drv_t * drv, const char * path, lv_fs_mode_t mode);
```

path is path after the driver 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 write or read.

The return value is a pointer to the *file object* the 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. (see below)

path 是驱动程序字母后的路径（例如“S:path/to/file.txt” -> “path/to/file.txt”）。**mode** 可以是 **LV_FS_MODE_WR** 或 **LV_FS_MODE_RD** 来打开写入或读取。

返回值是 *file object* 的指针，它描述了打开的文件，如果有任何问题（例如找不到文件），则返回“NULL”。返回的文件对象将传递给其他与文件系统相关的回调。（见下文）

Other callbacks (其他回调)

The other callbacks are quite similar. For example `write_cb` looks like this:

其他回调非常相似。例如 `write_cb` 看起来像这样：

```
lv_fs_res_t (*write_cb)(lv_fs_drv_t *drv, void *file_p, const void *buf, uint32_t
↪btw, uint32_t *bw);
```

As `file_p` LVGL passes the return value of `open_cb`, `buf` is the data to write, `btw` is the Bytes To Write, `bw` is the actually written bytes.

For a template to the callbacks see [lv_fs_template.c](#).

由于 `file_p` LVGL 传递 `open_cb` 的返回值, `buf` 是要写入的数据, `btw` 是要写入的字节数, `bw` 是实际写入的字节数。

有关回调的模板, 请参阅 [lv_fs_template.c](#)。

5.13.3 Usage example (使用示例)

The example below shows how to read from a file:

下面的示例演示如何从文件中读取：

```
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 write 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 the directories, but it can be a good practice to insert a '/' in front of the directory name.

`lv_fs_open` 中的模式可以是 `LV_FS_MODE_WR` to open for write 或 `LV_FS_MODE_RD` | `LV_FS_MODE_WR` 为两者

此示例演示如何读取目录的内容。如何标记目录取决于驱动程序, 但在目录名称前插入 “/” 是一个好习惯。

```
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);
    if(res != LV_FS_RES_OK) {
        my_error_handling();
        break;
```

(下页继续)

(续上页)

```

}

/*fn is empty, if not more files to read*/
if(strlen(fn) == 0) {
    break;
}

printf("%s\n", fn);
}

lv_fs_dir_close(&dir);

```

5.13.4 Use drivers for images (使用图像驱动程序)

Image objects can be opened from files too (besides variables stored in the flash).

To use files in image widgets the following callbacks are required:

- open
- close
- read
- seek
- tell

Image 对象也可以从文件中打开（除了存储在闪存中的变量）。

要在图像小部件中使用文件，需要以下回调：

- 打开
- 关
- 读
- 寻找
- 告诉

5.13.5 API

Typedefs

```

typedef uint8_t lv_fs_res_t
typedef uint8_t lv_fs_mode_t
typedef uint8_t lv_fs_whence_t
typedef struct _lv_fs_drv_t lv_fs_drv_t

```

Enums

enum [anonymous]

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 [anonymous]

File open mode.

Values:

enumerator **LV_FS_MODE_WR**
enumerator **LV_FS_MODE_RD**

enum [anonymous]

Seek modes.

Values:

enumerator **LV_FS_SEEK_SET**
enumerator **LV_FS_SEEK_CUR**
enumerator **LV_FS_SEEK_END**

Functions

`void _lv_fs_init(void)`

Initialize the File system interface

`void lv_fs_drv_init(lv_fs_drv_t *drv)`

Initialize a file system driver with default values. It is used to surely have known values in the fields and not memory junk. After it you can set the fields.

参数 `drv` -- pointer to driver variable to initialize

`void lv_fs_drv_register(lv_fs_drv_t *drv_p)`

Add a new drive

参数 `drv_p` -- pointer to an `lv_fs_drv_t` structure which is init 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

参数 `letter` -- the driver letter

返回 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.

参数 `letter` -- letter of the drive

返回 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

参数

- `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`

返回 `LV_FS_RES_OK` or any error from `lv_fs_res_t` enum

`lv_fs_res_t lv_fs_close(lv_fs_file_t *file_p)`

Close an already opened file

参数 `file_p` -- pointer to a `lv_fs_file_t` variable

返回 `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

参数

- `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.

返回 `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

参数

- **file_p** -- pointer to a *lv_fs_file_t* variable
- **buf** -- pointer to a buffer with the bytes to write
- **btr** -- Bytes To Write
- **br** -- the number of real written bytes (Bytes Written). NULL if unused.

返回 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

参数

- **file_p** -- pointer to a *lv_fs_file_t* variable
- **pos** -- the new position expressed in bytes index (0: start of file)

返回 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

参数

- **file_p** -- pointer to a *lv_fs_file_t* variable
- **pos_p** -- pointer to store the position of the read write pointer

返回 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

参数

- **raddir_p** -- pointer to a '*lv_fs_dir_t*' variable
- **path** -- path to a directory

返回 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)

Read the next filename form a directory. The name of the directories will begin with '/'

参数

- **raddir_p** -- pointer to an initialized 'fs_dir_t' variable
- **fn** -- pointer to a buffer to store the filename

返回 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

参数 **raddir_p** -- pointer to an initialized 'fs_dir_t' variable

返回 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

参数 buf -- buffer to store the letters ('\0' added after the last letter)

返回 the buffer

const char ***lv_fs_get_ext**(const char *fn)

Return with the extension of the filename

参数 fn -- string with a filename

返回 pointer to the beginning extension or empty string if no extension

char ***lv_fs_up**(char *path)

Step up one level

参数 path -- pointer to a file name

返回 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)

参数 path -- pointer to a file name

返回 pointer to the beginning of the last element in the path

struct **_lv_fs_drv_t**

Public Members

char **letter**

bool (***ready_cb**)(struct **_lv_fs_drv_t** *drv)

void *(***open_cb**)(struct **_lv_fs_drv_t** *drv, const char *path, **lv_fs_mode_t** mode)

lv_fs_res_t (***close_cb**)(struct **_lv_fs_drv_t** *drv, void *file_p)

lv_fs_res_t (***read_cb**)(struct **_lv_fs_drv_t** *drv, void *file_p, void *buf, uint32_t btr, uint32_t *br)

lv_fs_res_t (***write_cb**)(struct **_lv_fs_drv_t** *drv, void *file_p, const void *buf, uint32_t btw, uint32_t *bw)

lv_fs_res_t (***seek_cb**)(struct **_lv_fs_drv_t** *drv, void *file_p, uint32_t pos, **lv_fs_whence_t** whence)

lv_fs_res_t (***tell_cb**)(struct **_lv_fs_drv_t** *drv, void *file_p, uint32_t *pos_p)

void *(***dir_open_cb**)(struct **_lv_fs_drv_t** *drv, const char *path)

lv_fs_res_t (***dir_read_cb**)(struct **_lv_fs_drv_t** *drv, void *raddir_p, char *fn)

lv_fs_res_t (***dir_close_cb**)(struct **_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
struct lv_fs_dir_t
```

Public Members

```
void *dir_d
lv_fs_drv_t *drv
```

5.14 Animations (动画)

You can automatically change the value of a variable between a start and an end value using animations. The animation will happen by periodically calling an "animator" function with the corresponding value parameter.

The *animator* functions have the following prototype:

您可以使用动画在开始值和结束值之间自动更改变量的值。动画将通过使用相应的 `value` 参数定期调用“animator”函数来发生。

animator 函数具有以下原型：

```
void func(void * var, lv_anim_var_t value);
```

This prototype is compatible with the majority of the *set* functions of LVGL. For example `lv_obj_set_x(obj, value)` or `lv_obj_set_width(obj, value)`

该原型与 LVGL 的大多数 *set* 函数兼容。例如 `lv_obj_set_x(obj, value)` 或 `lv_obj_set_width(obj, value)`

5.14.1 Create an animation (创建动画)

To create an animation an `lv_anim_t` variable has to be initialized and configured with `lv_anim_set_...()` functions.

要创建动画，必须使用 `lv_anim_set_...()` 函数初始化和配置 `lv_anim_t` 变量。

```
/* INITIALIZE AN ANIMATION
*-----*/
lv_anim_t a;
lv_anim_init(&a);

/* MANDATORY SETTINGS
*-----*/
/*Set the "animator" function*/
lv_anim_set_exec_cb(&a, (lv_anim_exec_xcb_t) lv_obj_set_x);
```

(下页继续)

(续上页)

```

/*Set the "animator" function*/
lv_anim_set_var(&a, obj);

/*Length of the animation [ms]*/
lv_anim_set_time(&a, duration);

/*Set start and end values. E.g. 0, 150*/
lv_anim_set_values(&a, start, end);

/* OPTIONAL SETTINGS
*-----*/

/*Time to wait before starting the animation [ms]*/
lv_anim_set_delay(&a, delay);

/*Set path (curve). Default is linear*/
lv_anim_set_path(&a, lv_anim_path_ease_in);

/*Set a callback to call when animation is ready.*/
lv_anim_set_ready_cb(&a, ready_cb);

/*Set a callback to call when animation is started (after delay).*/
lv_anim_set_start_cb(&a, start_cb);

/*Play the animation backward too with this duration. Default is 0 (disabled) [ms]*/
lv_anim_set_playback_time(&a, wait_time);

/*Delay before playback. Default is 0 (disabled) [ms]*/
lv_anim_set_playback_delay(&a, wait_time);

/*Number of repetitions. Default is 1. LV_ANIM_REPEAT_INFINITE for infinite
→repetition*/
lv_anim_set_repeat_count(&a, wait_time);

/*Delay before repeat. Default is 0 (disabled) [ms]*/
lv_anim_set_repeat_delay(&a, wait_time);

/*true (default): apply the start vale immediately, false: apply start vale after
→delay when then anim. really starts. */
lv_anim_set_early_apply(&a, true/false);

/* START THE ANIMATION
*-----*/
lv_anim_start(&a);                                     /*Start the animation*/

```

You can apply multiple different animations on the same variable at the same time. For example, animate the x and y coordinates with `lv_obj_set_x` and `lv_obj_set_y`. However, only one animation can exist with a given variable and function pair. Therefore `lv_anim_start()` will delete the already existing variable-function animations.

您可以同时对同一个变量应用多个不同的动画。例如，使用 `lv_obj_set_x` 和 `lv_obj_set_y` 为 x 和 y 坐标设置动画。但是，对于给定的变量和函数对，只能存在一个动画。因此 `lv_anim_start()` 将删除已经存在的可变函数动画。

5.14.2 Animation path (动画轨迹)

You can determinate the path of animation. The most simple case is linear, meaning the current value between *start* and *end* is changed with fixed steps. A *path* is a function which calculates the next value to set based on the current state of the animation. Currently, there are the following built-in paths functions:

- `lv_anim_path_linear` linear animation
- `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 at the 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)

您可以确定动画的路径。最简单的情况是线性的，这意味着 *start* 和 *end* 之间的当前值以固定步长变化。*path* 是一个函数，它根据动画的当前状态计算要设置的下一个值。目前，有以下内置路径函数：

- `lv_anim_path_linear` 线性动画
- `lv_anim_path_step` 最后一步改变
- `lv_anim_path_ease_in` 开始时很慢
- `lv_anim_path_ease_out` 最后慢
- `lv_anim_path_ease_in_out` 开始和结束都很慢
- `lv_anim_path_overshoot` 超过结束值
- `lv_anim_path_bounce` 从最终值反弹一点（比如撞墙）

5.14.3 Speed vs time (速度与时间)

By default, you set the animation time. But in some cases, setting the animation speed is more practical.

The `lv_anim_speed_to_time(speed, start, end)` function calculates the required time in milliseconds to reach the end value from a start value with the given speed. The speed is interpreted in *unit/sec* dimension. For example, `lv_anim_speed_to_time(20, 0, 100)` will yield 5000 milliseconds. For example, in case of `lv_obj_set_x` *unit* is pixels so 20 means 20 *px/sec* speed.

默认情况下，您设置动画时间。但在某些情况下，设置动画速度更实用。

`lv_anim_speed_to_time(speed, start, end)` 函数计算从给定速度的起始值到达结束值所需的时间（以毫秒为单位）。速度以 *unit/sec* 维度解释。例如，`lv_anim_speed_to_time(20, 0, 100)` 将产生 5000 毫秒。例如，在 `lv_obj_set_x` 的情况下 *unit* 是像素，所以 20 意味着 20 *px/sec* 速度。

5.14.4 Delete animations (删除动画)

You can delete an animation with `lv_anim_del(var, func)` if you provide the animated variable and its animator function.

如果您提供动画变量及其动画器函数，您可以使用 `lv_anim_del(var, func)` 删除动画。

5.14.5 Timeline (时间线)

Timeline is a collection of multiple Animations, which makes it easy to create complex composite animations.

Firstly, create the animation element, but don't call `lv_anim_start()`.

Secondly, create an animation timeline object, by calling `lv_anim_timeline_create()`.

Thirdly, add animation elements to the animation timeline, by calling `lv_anim_timeline_add(at, start_time, &a)`. `start_time` is the start time of the animation on the timeline. Note that `start_time` will override the value of `delay`.

Finally, call `lv_anim_timeline_start(at)` to start the animation timeline.

时间线是多个动画的集合，可以轻松创建复杂的复合动画。

首先，创建动画元素，但不要调用 `lv_anim_start()`。

其次，通过调用 `lv_anim_timeline_create()` 创建一个动画时间线对象。

第三，通过调用 `lv_anim_timeline_add(at, start_time, &a)` 将动画元素添加到动画时间线。`start_time` 是时间线上动画的开始时间。请注意，`start_time` 将覆盖 `delay` 的值。

最后，调用 `lv_anim_timeline_start(at)` 启动动画时间线。

It supports forward and backward playback of the entire animation group, using `lv_anim_timeline_set_reverse(at, reverse)`.

Call the `lv_anim_timeline_set_progress(at, progress)` function to set the state of the object corresponding to the progress of the timeline.

Call the `lv_anim_timeline_get_playtime(at)` function to get the total duration of the entire animation timeline.

Call the `lv_anim_timeline_get_reverse(at)` function to get whether to reverse the animation timeline.

Call the `lv_anim_timeline_del(at)` function to delete the animation timeline.

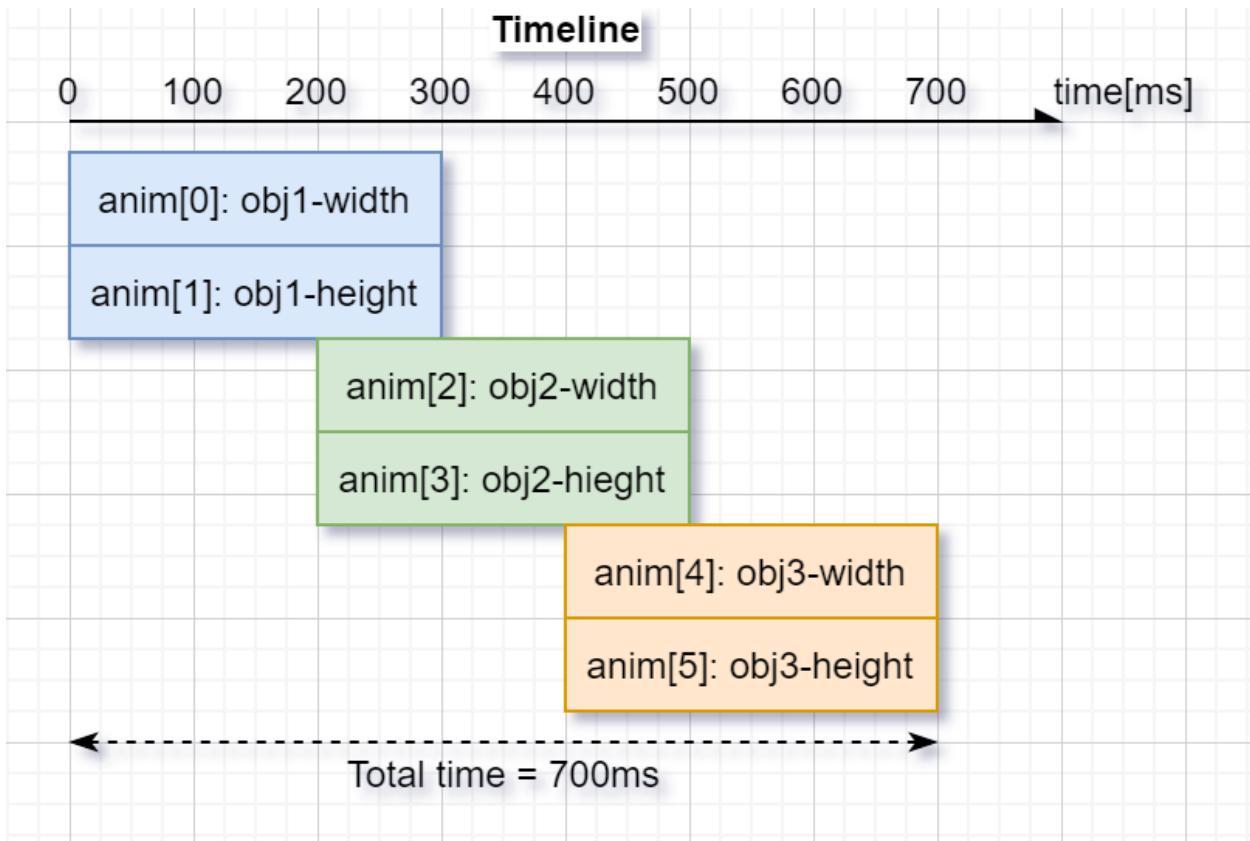
它支持整个动画组的向前和向后播放，使用 `lv_anim_timeline_set_reverse(at, reverse)`。

调用 `lv_anim_timeline_set_progress(at, progress)` 函数设置时间线进度对应的对象状态。

调用 `lv_anim_timeline_get_playtime(at)` 函数获取整个动画时间线的总时长。

调用 `lv_anim_timeline_get_reverse(at)` 函数获取是否反转动画时间线。

调用 `lv_anim_timeline_del(at)` 函数删除动画时间线。



5.14.6 Examples

5.14.7 API

TypeDefs

```
typedef int32_t (*lv_anim_path_cb_t)(const struct 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 its 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)(struct 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_ready_cb_t)(struct lv_anim_t*)
Callback to call when the animation is ready
```

```
typedef void (*lv_anim_start_cb_t)(struct lv_anim_t*)
Callback to call when the animation really stars (considering delay)
```

typedef int32_t (***lv_anim_get_value_cb_t**)(struct *lv_anim_t**)
 Callback used when the animation values are relative to get the current value

typedef struct *lv_anim_t* **lv_anim_t**
 Describes an animation

Enums

enum **lv_anim_enable_t**
 Can be used to indicate if animations are enabled or disabled in a case

Values:

enumerator **LV_ANIM_OFF**
 enumerator **LV_ANIM_ON**

Functions

LV_EXPORT_CONST_INT(LV_ANIM_REPEAT_INFINITE)

void **_lv_anim_core_init**(void)
 Init. the animation module

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);`
 参数 **a** -- pointer to an **lv_anim_t** variable to initialize

static inline void **lv_anim_set_var**(*lv_anim_t* ***a**, void ***var**)
 Set a variable to animate

参数

- **a** -- pointer to an initialized **lv_anim_t** variable
- **var** -- pointer to a variable to animate

static inline void **lv_anim_set_exec_cb**(*lv_anim_t* ***a**, *lv_anim_exec_xcb_t* **exec_cb**)
 Set a function to animate **var**

参数

- **a** -- pointer to an initialized **lv_anim_t** variable
- **exec_cb** -- a function to execute during animation LittlevGL's built-in functions can be used.
 E.g. `lv_obj_set_x`

static inline void **lv_anim_set_time**(*lv_anim_t* ***a**, uint32_t **duration**)
 Set the duration of an animation

参数

- **a** -- pointer to an initialized **lv_anim_t** variable
- **duration** -- duration of the animation in milliseconds

static inline void **lv_anim_set_delay**(*lv_anim_t* ***a**, uint32_t **delay**)
 Set a delay before starting the animation

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay before the animation in milliseconds

static inline void **lv_anim_set_values**(`lv_anim_t` *a, int32_t start, int32_t end)

Set the start and end values of an animation

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **start** -- the start value
- **end** -- the end value

static inline 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 binded to other languages because it's more consistent to have `lv_anim_t *` as first parameter. The variable to animate can be stored in the animation's `user_sata`

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **exec_cb** -- a function to execute.

static inline void **lv_anim_set_path_cb**(`lv_anim_t` *a, `lv_anim_path_cb_t` path_cb)

Set the path (curve) of the animation.

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **path_cb** -- a function the get the current value of the animation.

static inline void **lv_anim_set_start_cb**(`lv_anim_t` *a, `lv_anim_ready_cb_t` start_cb)

Set a function call when the animation really starts (considering `delay`)

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **start_cb** -- a function call when the animation starts

static inline 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 the the returned current value.

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **get_value_cb** -- a function call when the animation starts

static inline void **lv_anim_set_ready_cb**(`lv_anim_t` *a, `lv_anim_ready_cb_t` ready_cb)

Set a function call when the animation is ready

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **ready_cb** -- a function call when the animation is ready

static inline void **lv_anim_set_playback_time**(`lv_anim_t` *a, uint32_t time)

Make the animation to play back to when the forward direction is ready

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **time** -- the duration of the playback animation in milliseconds. 0: disable playback

`static inline void lv_anim_set_playback_delay(lv_anim_t *a, uint32_t delay)`

Make the animation to play back to when the forward direction is ready

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay in milliseconds before starting the playback animation.

`static inline void lv_anim_set_repeat_count(lv_anim_t *a, uint16_t cnt)`

Make the animation repeat itself.

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **cnt** -- repeat count or `LV_ANIM_REPEAT_INFINITE` for infinite repetition. 0: to disable repetition.

`static inline void lv_anim_set_repeat_delay(lv_anim_t *a, uint32_t delay)`

Set a delay before repeating the animation.

参数

- **a** -- pointer to an initialized `lv_anim_t` variable
- **delay** -- delay in milliseconds before repeating the animation.

`static inline 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.

参数

- **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_start(lv_anim_t *a)`

Create an animation

参数 **a** -- an initialized 'anim_t' variable. Not required after call.

`static inline uint32_t lv_anim_get_delay(lv_anim_t *a)`

Get a delay before starting the animation

参数 **a** -- pointer to an initialized `lv_anim_t` variable

返回 delay before the animation in milliseconds

`bool lv_anim_del(void *var, lv_anim_exec_xcb_t exec_cb)`

Delete an animation of a variable with a given animator function

参数

- **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'

返回 true: at least 1 animation is deleted, false: no animation is deleted

void lv_anim_del_all(void)

Delete all the animations animation

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`.

参数

- **var** -- pointer to variable
- **exec_cb** -- a function pointer which is animating 'var', or NULL to delete all the animations of 'var'

返回 pointer to the animation.

static inline bool lv_anim_custom_del(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.

参数

- **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'

返回 true: at least 1 animation is deleted, false: no animation is deleted

uint16_t lv_anim_count_running(void)

Get the number of currently running animations

返回 the number of running animations

uint32_t lv_anim_speed_to_time(uint32_t speed, int32_t start, int32_t end)

Calculate the time of an animation with a given speed and the start and end values

参数

- **speed** -- speed of animation in unit/sec
- **start** -- start value of the animation
- **end** -- end value of the animation

返回 the required time [ms] for the animation with the given parameters

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

参数 `a` -- pointer to an animation

返回 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

参数 `a` -- pointer to an animation

返回 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

参数 a -- pointer to an animation

返回 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)

参数 a -- pointer to an animation

返回 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

参数 a -- pointer to an animation

返回 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

参数 a -- pointer to an animation

返回 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)

参数 a -- pointer to an animation

返回 the current value to set

`struct _lv_anim_t`

#include <lv_anim.h> Describes an animation

Public Members

`void *var`

Variable to animate

`lv_anim_exec_xcb_t exec_cb`

Function to execute to animate

`lv_anim_start_cb_t start_cb`

Call it when the animation is starts (considering `delay`)

`lv_anim_ready_cb_t ready_cb`

Call it when the animation is ready

`lv_anim_get_value_cb_t get_value_cb`

Get the current value in relative mode

`void *user_data`

Custom user data

`lv_anim_path_cb_t path_cb`

Describe the path (curve) of animations

int32_t start_value

Start value

int32_t current_value

Current value

int32_t end_value

End value

int32_t time

Animation time in ms

int32_t act_time

Current time in animation. Set to negative to make delay.

uint32_t playback_delay

Wait before play back

uint32_t playback_time

Duration of playback animation

uint32_t repeat_delay

Wait before repeat

uint16_t repeat_cnt

Repeat count for the animation

uint8_t early_apply

1: Apply start value immediately even is there is `delay`

uint8_t playback_now

Play back is in progress

uint8_t run_round

Indicates the animation has run in this round

uint8_t start_cb_called

Indicates that the `start_cb` was already called

uint32_t time_orig

5.15 Timers (定时器)

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 [Porting](#) for more information.

The timers are non-preemptive, which means a timer cannot interrupt another timer. Therefore, you can call any LVGL related function in a timer.

LVGL 有一个内置的定时器系统。您可以注册一个函数以定期调用它。定时器在 `lv_timer_handler()` 中被处理和调用，它需要每隔几毫秒调用一次。有关详细信息，请参阅[移植](#)。

定时器是非抢占式的，这意味着一个定时器不能中断另一个定时器。因此，您可以在计时器中调用任何与 LVGL 相关的函数。

5.15.1 Create a timer (创建定时器)

To create a new timer, use `lv_timer_create(timer_cb, period_ms, user_data)`. It will create an `lv_timer_t *` variable, which can be used later to modify the parameters of the timer. `lv_timer_create_basic()` can also be used. This allows you to create a new timer without specifying any parameters.

A timer callback should have `void (*lv_timer_cb_t)(lv_timer_t *)`; prototype.

For example:

要创建一个新的计时器，请使用 `lv_timer_create(timer_cb, period_ms, user_data)`。它将创建一个 `lv_timer_t *` 变量，以后可以使用它来修改定时器的参数。也可以使用 `lv_timer_create_basic()`。这允许您在不指定任何参数的情况下创建新计时器。

一个定时器回调应该有 `void (*lv_timer_cb_t)(lv_timer_t *)`; 原型。

例如：

```
void my_timer(lv_timer_t * timer)
{
    /*Use the user_data*/
    uint32_t * user_data = timer->user_data;
    printf("my_timer called with user data: %d\n", *user_data);

    /*Do something with LVGL*/
    if(something_happened) {
        something_happened = false;
        lv_btn_create(lv_scr_act(), NULL);
    }
}

...

static uint32_t user_data = 10;
lv_timer_t * timer = lv_timer_create(my_timer, 500, &user_data);
```

5.15.2 Ready and Reset (准备与重置)

`lv_timer_ready(timer)` makes the 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 the defined period of milliseconds has elapsed.

`lv_timer_ready(timer)` 使计时器在下一次调用 `lv_timer_handler()` 时运行。

`lv_timer_reset(timer)` 重置计时器的周期。它将在定义的毫秒时间段过去后再次调用。

5.15.3 Set parameters (参数设置)

You can modify some parameters of the timers later:

- `lv_timer_set_cb(timer, new_cb)`
- `lv_timer_set_period(timer, new_period)`

您可以修改定时器的一些参数:

- `lv_timer_set_cb(timer, new_cb)`
- `lv_timer_set_period(timer, new_period)`

5.15.4 Repeat count (设置重复次数)

You can make a timer repeat only a given number of times with `lv_timer_set_repeat_count(timer, count)`. The timer will automatically be deleted after being called the defined number of times. Set the count to -1 to repeat indefinitely.

您可以使用 `lv_timer_set_repeat_count(timer, count)` 使计时器仅重复给定次数。定时器在调用定义的次数后将自动删除。将计数设置为 “-1” 以无限重复。

5.15.5 Measure idle time (测量空闲时间)

You can get the idle percentage time of `lv_timer_handler` with `lv_timer_get_idle()`. Note that, it doesn't measure the idle time of the overall system, only `lv_timer_handler`. It can be misleading if you use an operating system and call `lv_timer_handler` in a timer, as it won't actually measure the time the OS spends in an idle thread.

你可以通过 `lv_timer_get_idle()` 获取 `lv_timer_handler` 的空闲百分比时间。请注意，它不测量整个系统的空闲时间，只测量 `lv_timer_handler`。如果您使用操作系统并在计时器中调用 `lv_timer_handler`，这可能会产生误导，因为它实际上不会测量操作系统在空闲线程中花费的时间。

5.15.6 Asynchronous calls (异步调用)

In some cases, you can't do an action immediately. For example, you can't delete an object because something else is still using it or you don't want to block the execution now. For these cases, `lv_async_call(my_function, data_p)` can be used to make `my_function` be called on the next call of `lv_timer_handler`. `data_p` will be passed to function when it's called. Note that, only the pointer of the data is saved so you need to ensure that the variable will be "alive" while the function is called. It can be static, global or dynamically allocated data.

For example:

在某些情况下，您无法立即执行操作。例如，您不能删除一个对象，因为其他东西仍在使用它，或者您现在不想阻止执行。对于这些情况，可以使用 `lv_async_call(my_function, data_p)` 使 `my_function`

在下一次调用 `lv_timer_handler` 时被调用。`data_p` 将在调用时传递给函数。请注意，仅保存数据的指针，因此您需要确保在调用函数时该变量将“处于活动状态”。它可以是静态、全局或动态分配的数据。

例如：

```
void my_screen_clean_up(void * scr)
{
    /*Free some resources related to `scr`*/
    /*Finally delete the screen*/
    lv_obj_del(scr);
}

...
/*Do somethings with the object on the current screen*/

/*Delete screen on next call of `lv_timer_handler`, so not now.*/
lv_async_call(my_screen_clean_up, lv_scr_act());

/*The screen is still valid so you can do other things with it*/
```

If you just want to delete an object, and don't need to clean anything up in `my_screen_cleanup`, you could just use `lv_obj_del_async`, which will delete the object on the next call to `lv_timer_handler`.

如果你只是想删除一个对象，而不需要在 `my_screen_cleanup` 中清理任何东西，你可以使用 `lv_obj_del_async`，它会在下次调用 `lv_timer_handler` 时删除该对象。

5.15.7 API

Typedefs

`typedef void (*lv_timer_cb_t)(struct _lv_timer_t*)`

Timers execute this type of functions.

`typedef struct _lv_timer_t lv_timer_t`

Descriptor of a lv_timer

Functions

`void _lv_timer_core_init(void)`
Init the lv_timer module

`lv_timer_t *lv_timer_create_basic(void)`
Create an "empty" timer. It needs to initialized with at least `lv_timer_set_cb` and `lv_timer_set_period`

返回 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

参数

- **timer_xcb** -- a callback to call periodically. (the 'x' in the argument name indicates that its 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

返回 pointer to the new timer

`void lv_timer_del(lv_timer_t *timer)`

Delete a lv_timer

参数 **timer** -- pointer to an lv_timer

`void lv_timer_pause(lv_timer_t *timer)`

Pause/resume a timer.

参数

- **timer** -- pointer to an lv_timer
- **pause** -- true: pause the timer; false: resume

`void lv_timer_resume(lv_timer_t *timer)`

`void lv_timer_set_cb(lv_timer_t *timer, lv_timer_cb_t timer_cb)`

Set the callback the timer (the function to call periodically)

参数

- **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

参数

- **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.

参数 **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.

参数

- **timer** -- pointer to a lv_timer.
- **repeat_count** -- -1 : infinity; 0 : stop ; n>0: residual times

`void lv_timer_reset(lv_timer_t *timer)`

Reset a lv_timer. It will be called the previously set period milliseconds later.

参数 **timer** -- pointer to a lv_timer.

`void lv_timer_enable(bool en)`

Enable or disable the whole lv_timer handling

参数 **en** -- true: lv_timer handling is running, false: lv_timer handling is suspended

```
uint8_t lv_timer_get_idle(void)
Get idle percentage
```

返回 the lv_timer idle in percentage

```
lv_timer_t *lv_timer_get_next(lv_timer_t *timer)
Iterate through the timers
```

参数 **timer** -- NULL to start iteration or the previous return value to get the next timer

返回 the next timer or NULL if there is no more timer

```
struct _lv_timer_t
#include <lv_timer.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

Typedefs

```
typedef void (*lv_async_cb_t)(void*)
```

Type for async callback.

Functions

```
lv_res_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!

参数

- **async_xcb** -- a callback which is the task itself. (the 'x' in the argument name indicates that its not a fully generic function because it not follows the `func_name(object, callback, ...)` convention)
- **user_data** -- custom parameter

5.16 Drawing (绘画)

With LVGL, you don't need to draw anything manually. Just create objects (like buttons, labels, arc, etc), move and change them, and LVGL will refresh and redraw what is required.

However, it might be useful to have a basic understanding of how drawing happens in LVGL to add customization, make it easier to find bugs or just out of curiosity.

The basic concept is to not draw directly to the screen, but draw to an internal draw buffer first. When drawing (rendering) is ready, that buffer is copied to the screen.

The draw buffer can be smaller than the screen's size. LVGL will simply render in "tiles" that fit into the given draw buffer.

使用 LVGL，您无需手动绘制任何内容。只需创建对象（如按钮、标签、圆弧等）、移动和更改它们，LVGL 将刷新并重绘所需的内容。

但是，对 LVGL 中的绘制方式有一个基本的了解以添加自定义、更容易地发现错误或只是出于好奇可能会很有用。

基本概念是不直接绘制到屏幕，而是首先绘制到内部绘制缓冲区。当绘图（渲染）准备好时，该缓冲区被复制到屏幕上。

绘制缓冲区可以小于屏幕的大小。LVGL 将简单地在适合给定绘制缓冲区的“图块”中进行渲染。

This approach has two main advantages compared to directly drawing to the screen:

1. It avoids flickering while the layers of the UI are drawn. For example, if LVGL drawn directly into the display, when drawing a *background + button + text*, each "stage" would be visible for a short time .
2. It's faster to modify a buffer in internal RAM and finally write one pixel only once than reading/writing the display directly on each pixel access. (e.g. via a display controller with SPI interface).

Note that this concept is different from "traditional" double buffering where there are 2 screen sized frame buffers: one holds the current image to show on the display, and rendering happens to the other (inactive) frame buffer, and they are swapped when the rendering is finished. The main difference is that with LVGL you don't have to store 2 frame buffers (which usually requires external RAM) but only smaller draw buffer(s) that can easily fit into the internal RAM too.

与直接绘制到屏幕相比，这种方法有两个主要优点：

1. 避免绘制 UI 层时闪烁。例如，如果 LVGL 直接绘制到显示中，那么在绘制 * 背景 + 按钮 + 文本 * 时，每个“阶段”都会在短时间内可见。
2. 修改内部 RAM 中的缓冲区并最终仅写入一个像素一次比在每个像素访问时直接读取/写入显示更快。（例如，通过带有 SPI 接口的显示控制器）。

请注意，此概念与“传统”双缓冲不同，后者有 2 个屏幕大小的帧缓冲区：一个保存当前图像以显示在显示器上，渲染发生在另一个（非活动）帧缓冲区中，渲染完成后它们会被交换。主要区别在于，使用 LVGL，您不必存储 2 个帧缓冲区（通常需要外部 RAM），而只需存储更小的绘图缓冲区，也可以轻松装入内部 RAM。

5.16.1 Mechanism of screen refreshing (屏幕刷新机制)

Be sure to get familiar with the *Buffering modes of LVGL* first.

LVGL refreshes the screen in the following steps:

1. Something happens on the UI which requires redrawing. For example, a button is pressed, a chart is changed, an animation happened, etc.
2. LVGL saves the changed object's old and new area into a buffer, called an *Invalid area buffer*. For optimization, in some cases, objects are not added to the buffer:

- Hidden objects are not added.
 - Objects completely out of their parent are not added.
 - Areas partially out of the parent are cropped to the parent's area.
 - The objects on other screens are not added.
3. In every `LV_DISP_DEF_REFR_PERIOD` (set in `lv_conf.h`) the followings happen:
- LVGL checks the invalid areas and joins the adjacent or intersecting areas.
 - Takes the first joined area, if it's smaller than the *draw buffer*, then simply render the area's content into the *draw buffer*. If the area doesn't fit into the buffer, draw as many lines as possible to the *draw buffer*.
 - When the area is rendered, call `flush_cb` from the display driver to refresh the display.
 - If the area was larger than the buffer, render the remaining parts too.
 - Do the same with all the joined areas.

一定要先熟悉LVGL 的缓冲机制。

LVGL 按以下步骤刷新屏幕：

1. UI 上发生了一些需要重绘的事情。例如，按下按钮、更改图表、发生动画等。
2. LVGL 将改变对象的旧区和新区保存到一个缓冲区中，称为无效区缓冲区。为了优化，在某些情况下，不会将对象添加到缓冲区中：
 - 不添加隐藏对象。
 - 不添加完全脱离其父对象的对象。
 - 部分超出父级的区域被裁剪到父级的区域。
 - 不添加其他屏幕上的对象。
3. 在每个 `LV_DISP_DEF_REFR_PERIOD` (在 `lv_conf.h` 中设置) 发生以下情况：
 - LVGL 检查无效区域并连接相邻或相交的区域。
 - 获取第一个连接区域，如果它小于 *draw buffer*，则只需将该区域的内容渲染到 *draw buffer* 中。如果该区域不适合缓冲区，则在 *draw buffer* 中绘制尽可能多的线。
 - 当区域被渲染时，从显示驱动程序调用 `flush_cb` 来刷新显示。
 - 如果该区域大于缓冲区，也渲染其余部分。
 - 对所有连接的区域执行相同操作。

When an area is redrawn, the library searches the top most object which covers that area, and starts drawing from that object. For example, if a button's label has changed, the library will see that it's enough to draw the button under the text, and that it's not required to draw the screen under the button too.

The difference between buffering modes regarding the drawing mechanism is the following:

1. **One buffer** - LVGL needs to wait for `lv_disp_flush_ready()` (called from `flush_cb`) before starting to redraw the next part.
2. **Two buffers** - LVGL can immediately draw to the second buffer when the first is sent to `flush_cb` because the flushing should be done by DMA (or similar hardware) in the background.
3. **Double buffering** - `flush_cb` should only swap the address of the frame buffer.

当一个区域被重绘时，库搜索覆盖该区域的最上面的对象，并从该对象开始绘制。例如，如果按钮的标签发生了变化，库将看到在文本下方绘制按钮就足够了，并且不需要在按钮下方绘制屏幕。

关于绘图机制的缓冲模式之间的区别如下：

1. **One buffer** - LVGL 需要等待 `lv_disp_flush_ready()` (从 `flush_cb` 调用), 然后才开始重绘下一部分。
2. **两个缓冲区** - 当第一个缓冲区发送到 `flush_cb` 时, LVGL 可以立即绘制到第二个缓冲区, 因为刷新应该由 DMA (或类似硬件) 在后台完成。
3. **双缓冲** - `flush_cb` 应该只交换帧缓冲区的地址。

5.16.2 Masking (蒙版)

Masking is the basic concept of LVGL's draw engine. To use LVGL it's not required to know about the mechanisms described here, but you might find interesting to know how drawing works under hood. Knowing about masking comes in handy if you want to customize drawing.

To learn masking let's learn the steps of drawing first. LVGL performs the following steps to render any shape, image or text. It can be considered as a drawing pipeline.

1. **Prepare the draw descriptors** Create a draw descriptor from an object's styles (e.g. `lv_draw_rect_dsc_t`). This gives us the parameters for drawing, for example the colors, widths, opacity, fonts, radius, etc.
2. **Call the draw function** Call the draw function with the draw descriptor and some other parameters (e.g. `lv_draw_rect()`). It renders the primitive shape to the current draw buffer.
3. **Create masks** If the shape is very simple and doesn't require masks go to #5. Else create the required masks (e.g. a rounded rectangle mask)
4. **Calculate all the added mask.** It creates 0..255 values into a *mask buffer* with the "shape" of the created masks. E.g. in case of a "line mask" according to the parameters of the mask, keep one side of the buffer as it is (255 by default) and set the rest to 0 to indicate that this side should be removed.
5. **Blend a color or image** During blending masks (make some pixels transparent or opaque), blending modes (additive, subtractive, etc) and opacity are handled.

Masking 是 LVGL 绘图引擎的基本概念。要使用 LVGL, 不需要了解这里描述的机制, 但您可能会发现了解如何在引擎盖下绘制是很有趣的。如果您想自定义绘图, 了解蒙版会派上用场。

要学习蒙版, 让我们先学习绘图的步骤。LVGL 执行以下步骤来渲染任何形状、图像或文本。它可以被认为是一个绘图管道。

1. 准备绘制描述符根据对象的样式 (例如 `lv_draw_rect_dsc_t`) 创建绘制描述符。这为我们提供了绘图参数, 例如颜色、宽度、不透明度、字体、半径等。
2. 调用绘图函数使用绘图描述符和一些其他参数 (例如 `lv_draw_rect()`) 调用绘图函数。它将原始形状渲染到当前绘制缓冲区。
3. 创建蒙版如果形状非常简单并且不需要蒙版, 请转到 #5。否则创建所需的蒙版 (例如圆角矩形蒙版)
4. 计算所有添加的蒙版。它使用创建的蒙版的“形状”将 0..255 个值创建到 蒙版缓冲区中。例如。如果根据蒙版的参数出现“线蒙版”, 则将缓冲区的一侧保持原样 (默认为 255) 并将其余部分设置为 0 以指示应删除该侧。
5. 混合颜色或图像在混合蒙版 (使一些像素透明或不透明) 期间, 处理混合模式 (加法、减法等) 和不透明度。

LVGL has the following built-in mask types which can be calculated and applied real-time:

- **LV_DRAW_MASK_TYPE_LINE** Removes a side from a line (top, bottom, left or right). `lv_draw_line` uses 4 of it. Essentially, every (skew) line is bounded with 4 line masks by forming a rectangle.
- **LV_DRAW_MASK_TYPE_RADIUS** Removes the inner or outer parts of a rectangle which can have radius. It's also used to create circles by setting the radius to large value (`LV_RADIUS_CIRCLE`)

- **LV_DRAW_MASK_TYPE_ANGLE** Removes a circle sector. It is used by `lv_draw_arc` to remove the "empty" sector.
- **LV_DRAW_MASK_TYPE_FADE** Create a vertical fade (change opacity)
- **LV_DRAW_MASK_TYPE_MAP** The mask is stored in an array and the necessary parts are applied

LVGL 具有以下可以实时计算和应用的内置蒙版类型：

- **LV_DRAW_MASK_TYPE_LINE** 从一条线（顶部、底部、左侧或右侧）中删除一侧。`lv_draw_line` 使用了其中的 4 个。本质上，每条（倾斜）线都通过形成一个矩形以 4 个线掩模为界。
- **LV_DRAW_MASK_TYPE_RADIUS** 移除可以有半径的矩形的内部或外部部分。它还用于通过将半径设置为大值 (`LV_RADIUS_CIRCLE`) 来创建圆
- **LV_DRAW_MASK_TYPE_ANGLE** 删除一个圆形扇区。`lv_draw_arc` 使用它来删除“空”扇区。
- **LV_DRAW_MASK_TYPE_FADE** 创建垂直淡入淡出（改变不透明度）
- **LV_DRAW_MASK_TYPE_MAP** 蒙版存储在一个数组中，并应用必要的部分

Masks are used to create almost every basic primitives:

- **letters** Create a mask from the letter and draw a rectangle with the letter's color considering the mask.
- **line** Created from 4 "line masks", to mask out the left, right, top and bottom part of the line to get perfectly perpendicular line ending.
- **rounded rectangle** A mask is created real-time to add radius to the corners.
- **clip corner** To clip to overflowing content (usually children) on the rounded corners also a rounded rectangle mask is applied.
- **rectangle border** Same as a rounded rectangle, but inner part is masked out too.
- **arc drawing** A circle border is drawn, but an arc mask is applied too.
- **ARGB images** The alpha channel is separated into a mask and the image is drawn as a normal RGB image.

蒙版用于创建几乎所有基本图元：

- **字母** 根据字母创建一个蒙版，并根据蒙版用字母的颜色绘制一个矩形。
- **line** 由 4 个“线蒙版”创建，用于遮住线的左、右、上和下部分，以获得完美垂直的线尾。
- **圆角矩形** 实时创建蒙版以向角添加半径。
- **剪辑角落** 为了剪辑到圆角上的溢出内容（通常是儿童），还应用了圆角矩形蒙版。
- **矩形边框** 与圆角矩形相同，但内部部分也被屏蔽了。
- **圆弧绘制** 绘制了圆形边框，但也应用了圆弧蒙版。
- **ARGB 图像** alpha 通道被分离成一个蒙版，图像被绘制为一个普通的 RGB 图像。

5.16.3 Hook drawing (挂钩绘图)

Although widgets can be very well customized by styles there might be cases when something really custom is required. To ensure a great level of flexibility LVGL sends a lot events during drawing with parameters that tell what LVGL is about to draw. Some fields of these parameters can be modified to draw something else or any custom drawing can be added manually.

A good use case for it is the `Button matrix` widget. By default its buttons can be styled in different states but you can't style the buttons one by one. However, an event is sent for every button and you can for example tell LVGL to use different colors on a specific button or to manually draw an image on some buttons.

Below each of these events are described in detail.

尽管小部件可以通过样式很好地自定义，但在某些情况下可能需要真正自定义。为了确保高度的灵活性，LVGL 在绘制过程中发送了很多事件，并带有告诉 LVGL 将要绘制什么的参数。可以修改这些参数的某些字段以绘制其他内容，或者可以手动添加任何自定义绘图。

它的一个很好的用例是 *Button matrix* 小部件。默认情况下，它的按钮可以在不同状态下设置样式，但您不能一个一个地设置按钮样式。但是，每个按钮都会发送一个事件，例如，您可以告诉 LVGL 在特定按钮上使用不同的颜色或在某些按钮上手动绘制图像。下面详细描述了这些事件中的每一个。

Main drawing (主图)

These events are related to the actual drawing of the object. E.g. drawing of buttons, texts, etc happens here.

`lv_event_get_clip_area(event)` can be used to get the current clip area. The clip area is required in draw functions to make them draw only on a limited area.

这些事件与对象的实际绘制有关。例如。按钮、文本等的绘制发生在这里。

`lv_event_get_clip_area(event)` 可以用来获取当前的剪辑区域。绘制函数中需要剪辑区域，以便它们仅在有限的区域上绘制。

LV_EVENT_DRAW_MAIN_BEGIN

Sent before starting to draw an object. This is a good place to add masks manually. E.g. add a line mask that "removes" the right side of an object.

在开始绘制对象之前发送。这是手动添加蒙版的好地方。例如。添加一个“删除”对象右侧的线蒙版。

LV_EVENT_DRAW_MAIN

The actual drawing of the object happens in this event. E.g. a rectangle for a button is drawn here. First, the widgets' internal events are called to perform drawing and after that you can draw anything on top of them. For example you can add a custom text or an image.

对象的实际绘制发生在此事件中。例如。这里绘制了一个按钮的矩形。首先，调用小部件的内部事件来执行绘图，然后您可以在它们之上绘制任何内容。例如，您可以添加自定义文本或图像。

LV_EVENT_DRAW_MAIN_END

Called when the main drawing is finished. You can draw anything here as well and it's also good place to remove the masks created in `LV_EVENT_DRAW_MAIN_BEGIN`.

在主绘图完成时调用。你也可以在这里画任何东西，它也是删除在 `LV_EVENT_DRAW_MAIN_BEGIN` 中创建的蒙版的好地方。

Post drawing (后绘图)

Post drawing events are called when all the children of an object are drawn. For example LVGL use the post drawing phase to draw the scrollbars because they should be above all the children.

`lv_event_get_clip_area(event)` can be used to get the current clip area.

当绘制对象的所有子项时，会调用绘制后事件。例如，LVGL 使用后期绘制阶段来绘制滚动条，因为它们应该位于所有子项之上。`lv_event_get_clip_area(event)` 可以用来获取当前的剪辑区域。

LV_EVENT_DRAW_POST_BEGIN

Sent before starting the post draw phase. Masks can be added here too to mask out the post drawn content.

在开始绘制后阶段之前发送。也可以在此处添加遮罩以遮蔽后期绘制的内容。

LV_EVENT_DRAW_POST

The actual drawing should happen here.

实际绘图应该发生在这里。

LV_EVENT_DRAW_POST_END

Called when post drawing has finished. If the masks were not removed in `LV_EVENT_DRAW_MAIN_END` they should be removed here.

在后期绘制完成时调用。如果在 `LV_EVENT_DRAW_MAIN_END` 中没有移除遮罩，则应在此处移除。

Part drawing (零件绘图)

When LVGL draws a part of an object (e.g. a slider's indicator, a table's cell or a button matrix's button) it sends events before and after drawing that part with some context of the drawing. It allows changing the parts on a very low level with masks, extra drawing, or changing the parameters that LVGL is planning to use for drawing.

In these events an `lv_obj_draw_part_t` structure is used to describe the context of the drawing. Not all fields are set for every part and widget. To see which fields are set for a widget see the widget's documentation.

`lv_obj_draw_part_t` has the following fields:

当 LVGL 绘制对象的一部分（例如滑块的指示器、表格的单元格或按钮矩阵的按钮）时，它会在绘制该部分之前和之后使用绘图的某些上下文发送事件。它允许使用蒙版、额外绘图或更改 LVGL 计划用于绘图的参数在非常低的级别上更改零件。

在这些事件中，`lv_obj_draw_part_t` 结构用于描述绘图的上下文。并非为每个部件和小部件设置所有字段。要查看为小部件设置了哪些字段，请参阅小部件的文档。

`lv_obj_draw_part_t` 具有以下字段：

```
// Always set
const lv_area_t * clip_area;           // The current clip area, required if you need to
// draw something in the event
uint32_t part;                         // The current part for which the event is sent
uint32_t id;                           // The index of the part. E.g. a button's index
// on button matrix or table cell index.
```

(下页继续)

(续上页)

```

// Draw descriptors, set only if related
lv_draw_rect_dsc_t * rect_dsc;           // A draw descriptor that can be modified to
// changed what LVGL will draw. Set only for rectangle-like parts
lv_draw_label_dsc_t * label_dsc;          // A draw descriptor that can be modified to
// changed what LVGL will draw. Set only for text-like parts
lv_draw_line_dsc_t * line_dsc;            // A draw descriptor that can be modified to
// changed what LVGL will draw. Set only for line-like parts
lv_draw_img_dsc_t * img_dsc;              // A draw descriptor that can be modified to
// changed what LVGL will draw. Set only for image-like parts
lv_draw_arc_dsc_t * arc_dsc;              // A draw descriptor that can be modified to
// changed what LVGL will draw. Set only for arc-like parts

// Other parameters
lv_area_t * draw_area;                  // The area of the part being drawn
const lv_point_t * p1;                  // A point calculated during drawing. E.g. a
// point of chart or the center of an arc.
const lv_point_t * p2;                  // A point calculated during drawing. E.g. a
// point of chart.
char text[16];                         // A text calculated during drawing. Can be
// modified. E.g. tick labels on a chart axis.
lv_coord_t radius;                     // E.g. the radius of an arc (not the corner
// radius).
int32_t value;                         // A value calculated during drawing. E.g. Chart
// 's tick line value.
const void * sub_part_ptr;              // A pointer that identifies something in the part.
// E.g. chart series.

```

`lv_event_get_draw_part_dsc(event)` 可以用来获取一个指向 `lv_obj_draw_part_t` 的指针。

`lv_event_get_draw_part_dsc(event)` 可用于获取指向 `lv_obj_draw_part_t` 的指针。

LV_EVENT_DRAW_PART_BEGIN

开始绘制零件。这是修改绘图描述符（例如 `rect_dsc`）或添加遮罩的好地方。

LV_EVENT_DRAW_PART_END

Finish the drawing of a part. This is a good place to draw extra content on the part, or remove the masks added in `LV_EVENT_DRAW_PART_BEGIN`.

完成零件的绘制。这是在零件上绘制额外内容的好地方，或者删除在 `LV_EVENT_DRAW_PART_BEGIN` 中添加的蒙版。

Others (其他)

LV_EVENT_COVER_CHECK

This event is used to check whether an object fully covers an area or not.

`lv_event_get_cover_area(event)` returns an pointer to an area to check and `lv_event_set_cover_res(event, res)` can be used to set one of these results:

- `LV_COVER_RES_COVER` the areas is fully covered by the object
- `LV_COVER_RES_NOT_COVER` the areas is not covered by the object
- `LV_COVER_RES_MASKED` there is a mask on the object so it can not cover the area

Here are some reasons why an object would be unable to fully cover an area:

- It's simply not fully in area
- It has a radius
- It has not 100% background opacity
- It's an ARGB or chroma keyed image
- It does not have normal blending mode. In this case LVGL needs to know the colors under the object to do the blending properly
- It's a text, etc

此事件用于检查对象是否完全覆盖一个区域。

`lv_event_get_cover_area(event)` 返回一个指向要检查的区域的指针, `lv_event_set_cover_res(event, res)` 可用于设置以下结果之一:

- `LV_COVER_RES_COVER` 区域被对象完全覆盖
- `LV_COVER_RES_NOT_COVER` 区域未被对象覆盖
- `LV_COVER_RES_MASKED` 对象上有一个遮罩，所以它不能覆盖该区域

以下是物体无法完全覆盖区域的一些原因：

- 它只是不完全在区域内
- 它有一个半径
- 它没有 100% 的背景不透明度
- 这是一个 ARGB 或色度键控图像
- 它没有正常的混合模式。在这种情况下，LVGL 需要知道对象下的颜色才能正确进行混合
- 这是一个文本等

In short if for any reason the area below the object is visible than it doesn't cover that area.

Before sending this event LVGL checks if at least the widget's coordinates fully cover the area or not. If not the event is not called.

You need to check only the drawing you have added. The existing properties known by widget are handled in the widget's internal events. E.g. if a widget has > 0 radius it might not cover an area but you need to handle `radius` only if you will modify it and the widget can't know about it.

简而言之，如果由于某种原因对象下方的区域是可见的，则它不覆盖该区域。

在发送此事件之前，LVGL 检查至少小部件的坐标是否完全覆盖该区域。如果不是，则不会调用该事件。

您只需要检查您添加的图纸。小部件已知的现有属性在小部件的内部事件中处理。例如。如果小部件具有 > 0 半径，它可能不会覆盖一个区域，但只有当您修改它并且小部件无法知道它时，您才需要处理“半径”。

LV_EVENT_REFR_EXT_DRAW_SIZE

If you need to draw outside of a widget LVGL needs to know about it to provide the extra space for drawing. Let's say you create an event the writes the current value of a slider above its knob. In this case LVGL needs to know that the slider's draw area should be larger with the size required for the text.

You can simple set the required draw area with `lv_event_set_ext_draw_size(e, size)`.

如果您需要在小部件之外绘制，LVGL 需要了解它以提供额外的绘制空间。假设您创建了一个事件，将滑块的当前值写入其旋钮上方。在这种情况下，LVGL 需要知道滑块的绘制区域应该与文本所需的大小相同。

您可以使用 `lv_event_set_ext_draw_size(e, size)` 简单设置所需的绘图区域。

5.17 New widget

Widgets (部件)

6.1 Base object (基础对象) (`lv_obj`)

6.1.1 Overview

The 'Base Object' implements the basic properties of widgets on a screen, such as:

- coordinates
- parent object
- children
- contains the styles
- attributes like *Clickable*, *Scrollable*, etc.

In object-oriented thinking, it is the base class from which all other objects in LVGL are inherited.

The functions and functionalities of the Base object can be used with other widgets too. For example `lv_obj_set_width(slider, 100)`

The Base object can be directly used as a simple widget: it nothing else than a rectangle. In HTML terms, think of it as a `<div>`.

基础对象”实现了屏幕上小部件的基本属性，例如。

- 坐标
- 父对象
- 子对象
- 包含样式
- 属性，如可点击、可滚动等。

在面向对象的思维中，它是基类，LVGL 中的所有其他对象都是从它那里继承的。

Base 对象的功能和功能也可以与其他小部件一起使用。例如 `lv_obj_set_width(slider, 100)`

Base 对象可以直接用作一个简单的小部件：它只不过是一个矩形。在 HTML 术语中，将其视为。

Coordinates (坐标)

Only a small subset of coordinate settings is described here. To see all the features of LVGL (padding, coordinates in styles, layouts, etc) visit the [Coordinates](#) page.

此处仅描述了一小部分坐标设置。要查看 LVGL 的所有功能（填充、样式中的坐标、布局等），请访问[坐标](#)页面。

Size (大小)

The object size can be modified on individual axes with `lv_obj_set_width(obj, new_width)` and `lv_obj_set_height(obj, new_height)`, or both axes can be modified at the same time with `lv_obj_set_size(obj, new_width, new_height)`.

可以使用 `lv_obj_set_width(obj, new_width)` 和 `lv_obj_set_height(obj, new_height)` 修改单个轴上的对象大小，或者可以同时修改两个轴，使用高度 `lv_obj_set_width(obj,new_set_width)`。

Position (位置)

You can set the position relative to the parent with `lv_obj_set_x(obj, new_x)` and `lv_obj_set_y(obj, new_y)`, or both axes at the same time with `lv_obj_set_pos(obj, new_x, new_y)`.

您可以使用 `lv_obj_set_x(obj, new_x)` 和 `lv_obj_set_y(obj, new_y)` 设置相对于父级的位置，或者同时使用 `lv_obj_set_pos(obj, new_x), _new` 设置两个轴的位置。

Alignment (对齐)

You can align the object on its parent with `lv_obj_set_align(obj, LV_ALIGN_...)`. After this every x and y setting will be relative to the set alignment mode. For example a this will shift the object by 10;20 px from the center of its parent.

您可以使用 `lv_obj_set_align(obj, LV_ALIGN_...)` 将对象与其父对象对齐。此后，每个 x 和 y 设置都将适用于设置对齐模式。例如，这会将对象从其父项的中心移动 10;20 像素。

```
lv_obj_set_align(obj, LV_ALIGN_CENTER);
lv_obj_set_pos(obj, 10, 20);

//Or in one function
lv_obj_align(obj, LV_ALIGN_CENTER, 10, 20);
```

To align one object to another use `lv_obj_align_to(obj_to_align, obj_referece, LV_ALIGN_..., x, y)`

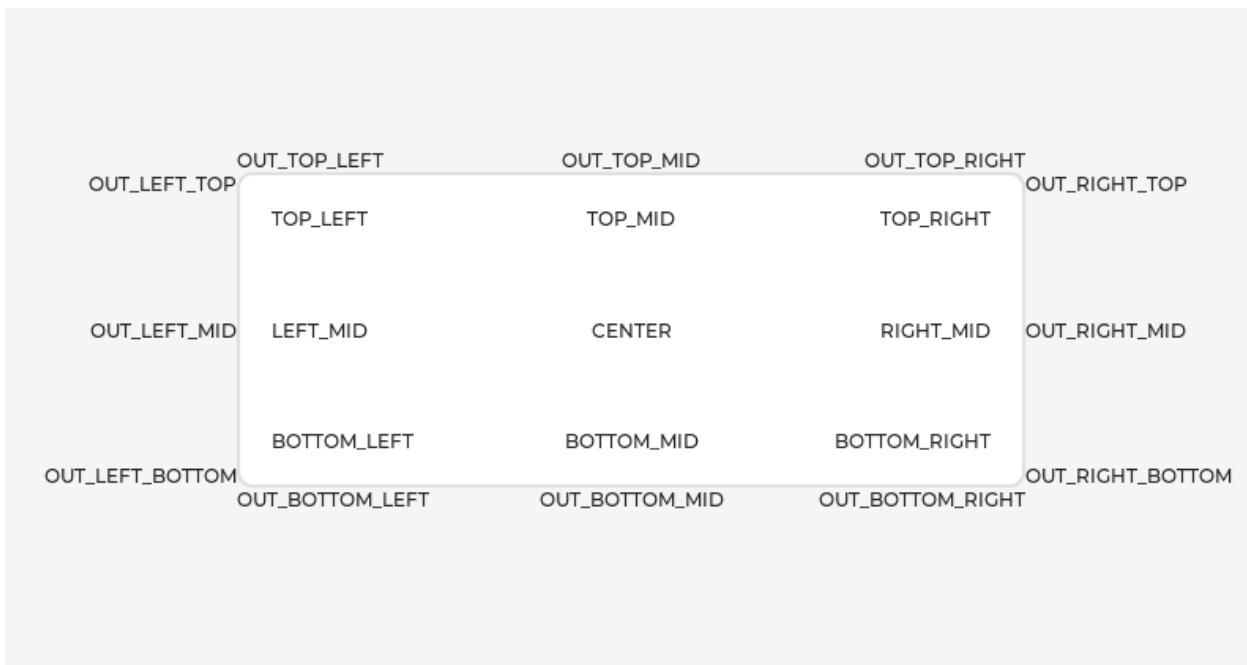
For example, to align a text below an image: `lv_obj_align(text, image, LV_ALIGN_OUT_BOTTOM_MID, 0, 10)`.

The following align types exist:

要将一个对象与另一个对象对齐，请使用 `lv_obj_align_to(obj_to_align, obj_referece, LV_ALIGN_..., x, y)`

例如，要对齐图片下方的文本：`lv_obj_align(text, image, LV_ALIGN_OUT_BOTTOM_MID, 0, 10)`。

存在以下对齐类型：



Parents and children (父母和孩子)

You can set a new parent for an object with `lv_obj_set_parent(obj, new_parent)`. To get the current parent, use `lv_obj_get_parent(obj)`.

To get a specific children of a parent use `lv_obj_get_child(parent, idx)`. Some examples for `idx`:

- 0 get the child created first child
- 1 get the child created second
- -1 get the child created last

The children can be iterated like this

您可以使用 `lv_obj_set_parent(obj, new_parent)` 为对象设置新的父级。要获取当前父级，请使用 `lv_obj_get_parent(obj)`。

要获取父母的特定孩子，请使用 `lv_obj_get_child(parent, idx)`。`idx` 的一些示例：

- 0 获取创建的第一个子项
- 1 获取创建的第二个子项
- -1 获取最后创建的子项

```
uint32_t i;
for(i = 0; i < lv_obj_get_child_cnt(parent); i++) {
    lv_obj_t * child = lv_obj_get_child(parent, i);
    /*Do something with child*/
}
```

`lv_obj_get_child_id(obj)` returns the index of the object. That is how many younger children its parent has.

You can bring an object to the foreground or send it to the background with `lv_obj_move_foreground(obj)` and `lv_obj_move_background(obj)`.

`lv_obj_get_child_id(obj)` 返回对象的索引。那就是它的父母有多少个年幼的孩子。

您可以使用 `lv_obj_move_foreground(obj)` 和 `lv_obj_move_background(obj)` 将对象带到前台或将其发送到后台。

Screens (屏幕)

When you have created a screen like `lv_obj_t * screen = lv_obj_create(NULL)`, you can load it with `lv_scr_load(screen)`. The `lv_scr_act()` function gives you a pointer to the current screen.

If you have multiple displays then it's important to know that these functions operate on the most-recently created or on the explicitly selected (with `lv_disp_set_default`) display.

To get an object's screen use the `lv_obj_get_screen(obj)` function.

当你创建了一个像 `lv_obj_t * screen = lv_obj_create(NULL)` 这样的屏幕后，你可以使用 `lv_scr_load(screen)` 加载它。`lv_scr_act()` 函数为您提供指向当前屏幕的指针。

如果您有多个显示器，那么重要的是要知道这些函数是在最近创建的还是在明确选择的（使用 `lv_disp_set_default`）显示器上运行的。

要获取对象的屏幕，请使用 `lv_obj_get_screen(obj)` 函数。

Events (事件)

To set an event callback for an object, use `lv_obj_add_event_cb(obj, event_cb, LV_EVENT_..., user_data)`,

To manually send an event to an object, use `lv_event_send(obj, LV_EVENT_..., param)`

Read the [Event overview](#) to learn more about the events.

要为对象设置事件回调，请使用 `lv_obj_add_event_cb(obj, event_cb, LV_EVENT_..., user_data)`，

要手动向对象发送事件，请使用 `lv_event_send(obj, LV_EVENT_..., param)`

阅读[事件概述](#)以了解有关事件的更多信息。

Styles (样式)

Be sure to read the [Style overview](#). Here only the most essential functions are described.

A new style can be added to an object with `lv_obj_add_style(obj, &new_style, selector)` function. `selector` is a combination of part and state(s). E.g. `LV_PART_SCROLLBAR | LV_STATE_PRESSED`.

The base objects use `LV_PART_MAIN` style properties and `LV_PART_SCROLLBAR` with the typical background style properties.

请务必阅读[样式概述](#)。这里只描述了最重要的功能。

可以使用 `lv_obj_add_style(obj, &new_style, selector)` 函数向对象添加新样式。选择器是部件和状态的组合。例如。`LV_PART_SCROLLBAR | LV_STATE_PRESSED`。

Flags (宏开关)

There are some attributes which can be enabled/disabled by `lv_obj_add/clear_flag(obj, LV_OBJ_FLAG_...)`:

- `LV_OBJ_FLAG_HIDDEN` Make the object hidden. (Like it wasn't there at all)
- `LV_OBJ_FLAG_CLICKABLE` Make the object clickable by the input devices
- `LV_OBJ_FLAG_CLICK_FOCUSABLE` Add focused state to the object when clicked
- `LV_OBJ_FLAG_CHECKABLE` Toggle checked state when the object is clicked
- `LV_OBJ_FLAG_SCROLLABLE` Make the object scrollable
- `LV_OBJ_FLAG_SCROLL_ELASTIC` Allow scrolling inside but with slower speed
- `LV_OBJ_FLAG_SCROLL_MOMENTUM` Make the object scroll further when "thrown"
- `LV_OBJ_FLAG_SCROLL_ONE` Allow scrolling only one snappable children
- `LV_OBJ_FLAG_SCROLL_CHAIN` Allow propagating the scroll to a parent
- `LV_OBJ_FLAG_SCROLL_ON_FOCUS` Automatically scroll object to make it visible when focused
- `LV_OBJ_FLAG_SNAPPABLE` If scroll snap is enabled on the parent it can snap to this object
- `LV_OBJ_FLAG_PRESS_LOCK` Keep the object pressed even if the press slid from the object
- `LV_OBJ_FLAG_EVENT_BUBBLE` Propagate the events to the parent too
- `LV_OBJ_FLAG_GESTURE_BUBBLE` Propagate the gestures to the parent
- `LV_OBJ_FLAG_ADV_HITTEST` Allow performing more accurate hit (click) test. E.g. consider rounded corners.
- `LV_OBJ_FLAG_IGNORE_LAYOUT` Make the object position-able by the layouts
- `LV_OBJ_FLAG_FLOATING` Do not scroll the object when the parent scrolls and ignore layout
- `LV_OBJ_FLAG_LAYOUT_1` Custom flag, free to use by layouts
- `LV_OBJ_FLAG_LAYOUT_2` Custom flag, free to use by layouts
- `LV_OBJ_FLAG_WIDGET_1` Custom flag, free to use by widget
- `LV_OBJ_FLAG_WIDGET_2` Custom flag, free to use by widget
- `LV_OBJ_FLAG_USER_1` Custom flag, free to use by user
- `LV_OBJ_FLAG_USER_2` Custom flag, free to use by user
- `LV_OBJ_FLAG_USER_3` Custom flag, free to use by user
- `LV_OBJ_FLAG_USER_4` Custom flag, free to use by usersection.

Some examples:

有一些属性可以通过 `lv_obj_add/clear_flag(obj, LV_OBJ_FLAG_...)` 启用/禁用:

- `LV_OBJ_FLAG_HIDDEN` 隐藏对象。(就像它根本不存在一样)
- `LV_OBJ_FLAG_CLICKABLE` 使输入设备可点击对象
- `LV_OBJ_FLAG_CLICK_FOCUSABLE` 单击时将焦点状态添加到对象
- `LV_OBJ_FLAG_CHECKABLE` 对象被点击时切换选中状态
- `LV_OBJ_FLAG_SCROLLABLE` 使对象可滚动
- `LV_OBJ_FLAG_SCROLL_ELASTIC` 允许在内部滚动但速度较慢

- LV_OBJ_FLAG_SCROLL_MOMENTUM 在“抛出”时使对象滚动得更远
- LV_OBJ_FLAG_SCROLL_ONE 只允许滚动一个可捕捉的孩子
- LV_OBJ_FLAG_SCROLL_CHAIN 允许将滚动传播到父级
- LV_OBJ_FLAG_SCROLL_ON_FOCUS 自动滚动对象以使其在聚焦时可见
- LV_OBJ_FLAG_SNAPPABLE 如果在父对象上启用了滚动捕捉，它可以捕捉到这个对象
- LV_OBJ_FLAG_PRESS_LOCK 保持对象被按下，即使按下从对象上滑动
- LV_OBJ_FLAG_EVENT_BUBBLE 也将事件传播给父级
- LV_OBJ_FLAG_GESTURE_BUBBLE 将手势传播给父级
- LV_OBJ_FLAG_ADV_HITTEST 允许执行更准确的命中（点击）测试。例如。考虑圆角。
- LV_OBJ_FLAG_IGNORE_LAYOUT 使对象可以通过布局定位
- LV_OBJ_FLAG_FLOATING 父滚动时不滚动对象，忽略布局
- LV_OBJ_FLAG_LAYOUT_1 自定义标志，可供布局免费使用
- LV_OBJ_FLAG_LAYOUT_2 自定义标志，可供布局免费使用
- LV_OBJ_FLAG_WIDGET_1 自定义标志，小部件免费使用
- LV_OBJ_FLAG_WIDGET_2 自定义标志，小部件免费使用
- LV_OBJ_FLAG_USER_1 自定义标志，用户免费使用
- LV_OBJ_FLAG_USER_2 自定义标志，用户免费使用
- LV_OBJ_FLAG_USER_3 自定义标志，用户免费使用
- LV_OBJ_FLAG_USER_4 自定义标志，由用户部分免费使用。一些例子：

```
/*Hide on object*/
lv_obj_add_flag(obj, LV_OBJ_FLAG_HIDDEN);

/*Make an obejct non-clickable*/
lv_obj_clear_flag(obj, LV_OBJ_FLAG_CLICKABLE);
```

Groups

Read the [Input devices overview](#) to learn more about the *Groups*.

Objects are added to a group with `lv_group_add_obj(group, obj)`, and you can use `lv_obj_get_group(obj)` to see which group an object belongs to.

`lv_obj_is_focused(obj)` returns if the object is currently focused on its group or not. If the object is not added to a group, `false` will be returned.

阅读[输入设备概述](#)以了解有关 *Groups* 的更多信息。

使用 `lv_group_add_obj(group, obj)` 将对象添加到组，您可以使用 `lv_obj_get_group(obj)` 查看对象属于哪个组。

`lv_obj_is_focused(obj)` 返回对象当前是否聚焦在其组上。如果对象未添加到组中，则将返回 `false`。

Extended click area (拓展的点击区域)

By default, the objects can be clicked only on their coordinates, however, this area can be extended with `lv_obj_set_ext_click_area(obj, size)`.

默认情况下，只能在对象的坐标上单击对象，但是，可以使用 `lv_obj_set_ext_click_area(obj, size)` 扩展该区域。

6.1.2 Events (事件)

- `LV_EVENT_VALUE_CHANGED` when the `LV_OBJ_FLAG_CHECKABLE` flag is enabled and the object clicked (on transition to/from the checked state)
- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` is sent for the following types:
 - `LV_OBJ_DRAW_PART_RECTANGLE` The main rectangle
 - * `part: LV_PART_MAIN`
 - * `rect_dsc`
 - * `draw_area`: the area of the rectangle
 - `LV_OBJ_DRAW_PART_BORDER_POST` The border if the `border_post` style property is `true`
 - * `part: LV_PART_MAIN`
 - * `rect_dsc`
 - * `draw_area`: the area of the rectangle
 - `LV_OBJ_DRAW_PART_SCROLLBAR` the scrollbars
 - * `part: LV_PART_SCROLLBAR`
 - * `rect_dsc`
 - * `draw_area`: the area of the rectangle

Learn more about [Events](#).

- `LV_EVENT_VALUE_CHANGED` 当 `LV_OBJ_FLAG_CHECKABLE` 标志被启用并且对象被点击时（转换到/从选中状态）
- 为以下类型发送 `LV_EVENT_DRAW_PART_BEGIN` 和 `LV_EVENT_DRAW_PART_END`:
 - `LV_OBJ_DRAW_PART_RECTANGLE` 主矩形
 - * 部分: `LV_PART_MAIN`-`rect_dsc`
 - * `draw_area`: 矩形的面积
 - `LV_OBJ_DRAW_PART_BORDER_POST` 如果 `border_post` 样式属性为 `true` 的边框
 - * 部分: `LV_PART_MAIN`-`rect_dsc`
 - * `draw_area`: 矩形的面积
 - `LV_OBJ_DRAW_PART_SCROLLBAR` 滚动条
 - * 部分: `LV_PART_SCROLLBAR`-`rect_dsc`
 - * `draw_area`: 矩形的面积

详细了解[事件](#)。

6.1.3 Keys

If `LV_OBJ_FLAG_CHECKABLE` is enabled `LV_KEY_RIGHT` and `LV_KEY_UP` make the object checked, and `LV_KEY_LEFT` and `LV_KEY_DOWN` make it unchecked.

Learn more about [Keys](#).

如果启用了 `LV_OBJ_FLAG_CHECKABLE`, 则 `LV_KEY_RIGHT` 和 `LV_KEY_UP` 使对象被选中, 而 `LV_KEY_LEFT` 和 `LV_KEY_DOWN` 使其不选中。

了解有关[Keys](#)的更多信息。

6.1.4 Example

C

Base objects with custom styles

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↳ lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_obj/lv_ex_obj_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↳ lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_obj/lv_ex_obj_1.py
```

MicroPython

No examples yet.

6.1.5 API

Typedefs

```
typedef uint16_t lv_state_t
typedef uint32_t lv_part_t
typedef uint32_t lv_obj_flag_t
typedef struct _lv_obj_t lv_obj_t
```

Enums

enum [**anonymous**]

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_SCROLLLED**

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 [anonymous]

The possible parts of widgets. The parts can be considered as the internal building block of the widgets. E.g. slider = background + indicator + knob Note every part is 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. tabel cells)

enumerator **LV_PART_TICKS**

Ticks on scale e.g. for a chart or meter

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 [anonymous]

On/Off features controlling the object's behavior. OR-ed values are possible

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 snapable children

enumerator **LV_OBJ_FLAG_SCROLL_CHAIN**

Allow propagating the scroll to a parent

enumerator **LV_OBJ_FLAG_SCROLL_ON_FOCUS**

Automatically scroll object to make it visible when focused

enumerator **LV_OBJ_FLAG_SNAPABLE**

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 position-able by the layouts

enumerator LV_OBJ_FLAG_FLOATING

Do not scroll the object when the parent scrolls and ignore layout

enumerator LV_OBJ_FLAG_LAYOUT_1**enumerator LV_OBJ_FLAG_LAYOUT_2**

Custom flag, free to use by layouts

enumerator LV_OBJ_FLAG_WIDGET_1

Custom flag, free to use by layouts

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 widget

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

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 Currently only implemented when not using custom allocators, or GC is enabled.

***lv_obj_t* *lv_obj_create(*lv_obj_t* *parent)**

Create a base object (a rectangle)

参数 parent -- pointer to a parent object. If NULL then a screen will be created.

返回 pointer to the new object

void lv_obj_add_flag(*lv_obj_t* *obj, *lv_obj_flag_t* f)

Set one or more flags

参数

- **obj** -- pointer to an object
- **f** -- R-ed values from *lv_obj_flag_t* to set.

void lv_obj_clear_flag(*lv_obj_t* *obj, *lv_obj_flag_t* f)

Clear one or more flags

参数

- **obj** -- pointer to an object
- **f** -- OR-ed values from *lv_obj_flag_t* to set.

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.

参数

- **obj** -- pointer to an object
- **state** -- the states to add. E.g *LV_STATE_PRESSED* | *LV_STATE_FOCUSED*

void lv_obj_clear_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.

参数

- **obj** -- pointer to an object
- **state** -- the states to add. E.g *LV_STATE_PRESSED* | *LV_STATE_FOCUSED*

void lv_obj_set_base_dir(*lv_obj_t* *obj, *lv_bidi_dir_t* dir)

Set the base direction of the object

参数

- **obj** -- pointer to an object
- **dir** -- the new base direction. *LV_BIDI_DIR_LTR/RTL/AUTO/INHERIT*

static inline void lv_obj_set_user_data(*lv_obj_t* *obj, void *user_data)

Set the user_data field of the object

参数

- **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.

参数

- **obj** -- pointer to an object
- **f** -- the flag(s) to check (OR-ed values can be used)

返回 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.

参数

- **obj** -- pointer to an object
- **f** -- the flag(s) to check (OR-ed values can be used)

返回 true: at least one flag is set; false: none of the flags are set

`lv_bidi_dir_t lv_obj_get_base_dir(const lv_obj_t *obj)`

Get the base direction of the object

参数 **obj** -- pointer to an object

返回 the base direction. LV_BIDI_DIR_LTR/RTL/AUTO/INHERIT

`lv_state_t lv_obj_get_state(const lv_obj_t *obj)`

Get the state of an object

参数 **obj** -- pointer to an object

返回 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.

参数

- **obj** -- pointer to an object
- **state** -- a state or combination of states to check

返回 true: obj is in state; false: obj is not in state

`void *lv_obj_get_group(const lv_obj_t *obj)`

Get the group of the object

参数 **obj** -- pointer to an object

返回 the pointer to group of the object

`static inline void *lv_obj_get_user_data(lv_obj_t *obj)`

Get the user_data field of the object

参数 **obj** -- pointer to an object

返回 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.

参数 **obj** -- pointer to an object

`bool lv_obj_check_type(const lv_obj_t *obj, const lv_obj_class_t *class_p)`

Get object's and its ancestors type. Put their name in `type_buf` starting with the current type. E.g. `buf.type[0] = "lv_btn", buf.type[1] = "lv_cont", buf.type[2] = "lv_obj"`

参数

- **obj** -- pointer to an object which type should be get
- **buf** -- pointer to an `lv_obj_type_t` buffer to store the types

`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.

参数

- **obj** -- pointer to an object
- **class_p** -- a class to check (e.g. `lv_slider_class`)

返回 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

参数 **obj** -- pointer to an object

返回 the class (type) of the object

bool lv_obj_is_valid(const *lv_obj_t* *obj)

Check if any object is still "alive", and part of the hierarchy

参数

- **obj** -- pointer to an object
- **obj_type** -- type of the object. (e.g. "lv_btn")

返回 true: valid

static inline lv_coord_t lv_obj_dpx(const *lv_obj_t* *obj, lv_coord_t n)

Scale the given number of pixels (a distance or size) relative to a 160 DPI display considering the DPI of the **obj**'s display. It ensures that e.g. `lv_dpx(100)` will have the same physical size regardless to the DPI of the display.

参数

- **obj** -- an object whose display's dpi should be considered
- **n** -- the number of pixels to scale

返回 $n \times \text{current_dpi}/160$

Variables

const *lv_obj_class_t* lv_obj_class

Make the base object's class publicly available.

struct lv_obj_spec_attr_t

#include <lv_obj.h> Special, rarely used attributes. They are allocated automatically if any elements is set.

Public Members

struct *lv_obj_t* **children

Store the pointer of the children in an array.

uint32_t child_cnt

Number of children

***lv_group_t* *group_p**

struct *lv_event_dsc_t* *event_dsc

Dynamically allocated event callback and user data array

lv_point_t scroll

The current X/Y scroll offset

lv_coord_t ext_click_pad

Extra click padding in all direction

lv_coord_t ext_draw_size

EXTend the size in every direction for drawing.

lv_scrollbar_mode_t scrollbar_mode

How to display scrollbars

lv_scroll_snap_t scroll_snap_x

Where to align the snapable children horizontally

lv_scroll_snap_t scroll_snap_y

Where to align the snapable children horizontally

lv_dir_t scroll_dir

The allowed scroll direction(s)

lv bidi_dir_t base_dir

Base direction of texts related to this object

uint8_t event_dsc_cnt

Number of event callbacks stored in event_cb array

struct **_lv_obj_t**

Public Members

const lv_obj_class_t ***class_p**

struct *_lv_obj_t* ***parent**

lv_obj_spec_attr_t ***spec_attr**

lv_obj_style_t ***styles**

void ***user_data**

lv_area_t **coords**

lv_obj_flag_t **flags**

lv_state_t **state**

uint16_t **layout_inv**

uint16_t **scr_layout_inv**

uint16_t **skip_trans**

uint16_t **style_cnt**

uint16_t **h_layout**

uint16_t **w_layout**

6.2 Core widgets (核心部件)

6.2.1 Arc (弧) (lv_arc)

Overview (概述)

The Arc consists of a background and a foreground arc. The foregrond (indicator) can be touch-adjusted.

弧由背景和前景弧组成。前景（指示器）可以进行触摸调整。

Parts and Styles

- LV_PART_MAIN Draws a background using the typical background style properties and an arc using the arc style properties. The arc's size and position will respect the *padding* style properties.
- LV_PART_INDICATOR Draws an other arc using the *arc* style properties. Its padding values are interpreted relative to the background arc.
- LV_PART_KNOB Draws a handle on the end of the indicator using all background properties and padding values. With zero padding the knob size is the same as the indicator's width. Larger padding makes it larger, smaller padding makes it smaller.
- LV_PART_MAIN 使用典型的背景样式属性绘制背景，使用圆弧样式属性绘制圆弧。圆弧的大小和位置将遵循 *padding* 样式属性。
- LV_PART_INDICATOR 使用 *arc* 样式属性绘制另一个圆弧。它的填充值是相对于背景弧来解释的。
- LV_PART_KNOB 使用所有背景属性和填充值在指标的末尾绘制一个句柄。使用零填充，旋钮大小与指示器的宽度相同。较大的填充使其更大，较小的填充使其更小。

Usage (用法)

Value and range (值和范围)

A new value can be set using `lv_arc_set_value(arc, new_value)`. The value is interpreted in a range (minimum and maximum values) which can be modified with `lv_arc_set_range(arc, min, max)`. The default range is 1..100.

The indicator arc is drawn on the main part's arc. This if the value is set to maximum the indicator arc will cover the entire "background" arc. To set the start and end angle of the background arc use the `lv_arc_set_bg_angles(arc, start_angle, end_angle)` functions or `lv_arc_set_bg_start/end_angle(arc, angle)`.

Zero degrees is at the middle right (3 o'clock) of the object and the degrees are increasing in clockwise direction. The angles should be in the [0;360] range.

可以使用 `lv_arc_set_value(arc, new_value)` 设置新值。该值在可以用 `lv_arc_set_range(arc, min, max)` 修改的范围（最小值和最大值）中解释。默认范围是 1..100。

指示弧绘制在主零件的弧上。如果该值设置为最大值，则指示器弧将覆盖整个“背景”弧。要设置背景弧的开始和结束角度，请使用 `lv_arc_set_bg_angles(arc, start_angle, end_angle)` 函数或 `lv_arc_set_bg_start/end_angle(arc, angle)`。

零度位于对象的中间右侧（3点钟方向），并且度数沿顺时针方向增加。角度应在 [0;360] 范围内。

Rotation (旋转)

An offset to the 0 degree position can be added with `lv_arc_set_rotation(arc, deg)`.

可以使用 `lv_arc_set_rotation(arc, deg)` 添加到 0 度位置的偏移量。

Mode (模式)

The arc can be one of the following modes:

- `LV_ARC_MODE_NORMAL` The indicator arc is drawn from the minimum value to the current.
- `LV_ARC_MODE_REVERSE` The indicator arc is drawn counter-clockwise from the maximum value to the current.
- `LV_ARC_MODE_SYMMETRICAL` The indicator arc is drawn from the middle point to the current value.

The mode can be set by `lv_arc_set_mode(arc, LV_ARC_MODE_...)` and used only if the angle is set by `lv_arc_set_value()` or the arc is adjusted by finger.

弧可以是以下模式之一：

- `LV_ARC_MODE_NORMAL` 指标弧从最小值绘制到当前值。
- `LV_ARC_MODE_REVERSE` 指标弧从最大值到当前值逆时针绘制。
- `LV_ARC_MODE_SYMMETRICAL` 指标弧从中间点绘制到当前值。

模式可以通过 `lv_arc_set_mode(arc, LV_ARC_MODE_...)` 设置，并且仅当角度由 `lv_arc_set_value()` 设置或通过手指调整弧度时使用。

Change rate (变化率)

If the arc is pressed the current value will be set with a limited speed according to the set *change rate*. The change rate is defined in degree/second unit and can be set with `lv_arc_set_change_rate(arc, rate)`

如果弧被按下，当前值将根据设置的变化率以有限的速度设置。变化率以度/秒为单位定义，可以用 `lv_arc_set_change_rate(arc, rate)` 设置

Setting the indicator manually (手动设置指示器)

It is also possible to set the angles of the indicator arc directly with `lv_arc_set_angles(arc, start_angle, end_angle)` function or `lv_arc_set_start/end_angle(arc, start_angle)`. In this case the set "value" and "mode" is ignored.

In other words, settings angles and values are independent. You should use either value and angle settings. Mixing the two might result in unintended behavior.

To make the arc non-adjustable remove the style of the knob and make the object non-clickable:

也可以直接使用 `lv_arc_set_angles(arc, start_angle, end_angle)` 函数或 `lv_arc_set_start/end_angle(arc, start_angle)` 设置指标弧的角度。在这种情况下，设置的“值”和“模式”将被忽略。

换句话说，设置角度和值是独立的。您应该使用值和角度设置。将两者混合可能会导致意外行为。

要使圆弧不可调整，请移除旋钮的样式并使对象不可点击：

```
lv_obj_remove_style(arc, NULL, LV_PART_KNOB);
lv_obj_clear_flag(arc, LV_OBJ_FLAG_CLICKABLE);
```

Events (事件)

- LV_EVENT_VALUE_CHANGED sent when the arc is pressed/dragged to set a new value.
- LV_EVENT_DRAW_PART_BEGIN and LV_EVENT_DRAW_PART_END are sent with the following types:
 - LV_ARC_DRAW_PART_BACKGROUND The background arc.
 - * part: LV_PART_MAIN
 - * p1: center of the arc
 - * radius: radius of the arc
 - * arc_dsc
 - LV_ARC_DRAW_PART_FOREGROUND The foreground arc.
 - * part: LV_PART_INDICATOR
 - * p1: center of the arc
 - * radius: radius of the arc
 - * arc_dsc
 - LV_ARC_DRAW_PART_KNOB The knob
 - * part: LV_PART_KNOB
 - * draw_area: the area of the knob
 - * rect_dsc:

See the events of the [Base object](#) too.

Learn more about [Events](#).

- 按下/拖动圆弧以设置新值时发送 “LV_EVENT_VALUE_CHANGED”。
- LV_EVENT_DRAW_PART_BEGIN 和 LV_EVENT_DRAW_PART_END 使用以下类型发送:
 - LV_ARC_DRAW_PART_BACKGROUND 背景弧。
 - * 部分: LV_PART_MAIN
 - * p1: 圆弧的中心
 - * radius: 弧的半径
 - * arc_dsc
 - LV_ARC_DRAW_PART_FOREGROUND 前景弧。
 - * 部分: LV_PART_INDICATOR
 - * p1: 圆弧的中心
 - * radius: 弧的半径
 - * arc_dsc
 - LV_ARC_DRAW_PART_KNOB 旋钮
 - * 部分: LV_PART_KNOB

* `draw_area`: 旋钮的面积 -`rect_dsc`:

参见[Base object](#) 的事件。

详细了解[事件](#)。

Keys (按键)

- `LV_KEY_RIGHT/UP` Increases the value by one.
- `LV_KEY_LEFT/DOWN` Decreases the value by one.

Learn more about [Keys](#).

- `LV_KEY_RIGHT/UP` 将值增加一。
- `LV_KEY_LEFT/DOWN` 将值减一。

了解有关[Keys](#) 的更多信息。

Example

C

Simple Arc

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

void lv_example_arc_1(void)
{
    /*Create an Arc*/
    lv_obj_t * arc = lv_arc_create(lv_scr_act());
    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, 40);
    lv_obj_center(arc);
}

#endif

# Create style for the Arcs
style = lv.style_t()
lv.style_copy(style, lv.style_plain)
style.line.color = lv.color_make(0,0,255) # Arc color
style.line.width = 8                      # Arc width

# Create an Arc
arc = lv.arc(lv.scr_act())
arc.set_style(lv.arc.STYLE.MAIN, style)    # Use the new style
arc.set_angles(90, 60)
arc.set_size(150, 150)
arc.align(None, lv.ALIGN.CENTER, 0, 0)
```

Loader with 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(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_scr_act());
    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_clear_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_time(&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
```

```
# Create an arc which acts as a loader.
class loader_arc(lv.arc):

    def __init__(self, parent, color=lv.color_hex(0x000080),
                 width=8, style=lv.style_plain, rate=20):
        super().__init__(parent)

        self.a = 0
        self.rate = rate

        # Create style for the Arcs
        self.style = lv.style_t()
        lv.style_copy(self.style, style)
        self.style.line.color = color
        self.style.line.width = width
```

(下页继续)

(续上页)

```

# Create an Arc
self.set_angles(180, 180);
self.set_style(self.STYLE.MAIN, self.style);

# Spin the Arc
self.spin()

def spin(self):
    # Create an `lv_task` to update the arc.
    lv.task_create(self.task_cb, self.rate, lv.TASK_PRIO.LOWEST, {})

    # An `lv_task` to call periodically to set the angles of the arc
def task_cb(self, task):
    self.a+=5;
    if self.a >= 359: self.a = 359

    if self.a < 180: self.set_angles(180-self.a, 180)
    else: self.set_angles(540-self.a, 180)

    if self.a == 359:
        self.a = 0
    lv.task_del(task)

# Create a loader arc
loader_arc = loader_arc(lv.scr_act())
loader_arc.align(None, lv.ALIGN.CENTER, 0, 0)

```

MicroPython

No examples yet.

API

TypeDefs

typedef uint8_t **lv_arc_mode_t**

Enums

enum [**anonymous**]

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 a arc objects

参数 **par** -- pointer to an object, it will be the parent of the new arc

返回 pointer to the created arc

`void lv_arc_set_start_angle(lv_obj_t *arc, uint16_t start)`

Set the start angle of an arc. 0 deg: right, 90 bottom, etc.

参数

- **arc** -- pointer to an arc object
- **start** -- the start angle

`void lv_arc_set_end_angle(lv_obj_t *arc, uint16_t end)`

Set the end angle of an arc. 0 deg: right, 90 bottom, etc.

参数

- **arc** -- pointer to an arc object
- **end** -- the end angle

`void lv_arc_set_angles(lv_obj_t *arc, uint16_t start, uint16_t end)`

Set the start and end angles

参数

- **arc** -- pointer to an arc object
- **start** -- the start angle
- **end** -- the end angle

`void lv_arc_set_bg_start_angle(lv_obj_t *arc, uint16_t start)`

Set the start angle of an arc background. 0 deg: right, 90 bottom, etc.

参数

- **arc** -- pointer to an arc object
- **start** -- the start angle

`void lv_arc_set_bg_end_angle(lv_obj_t *arc, uint16_t end)`

Set the start angle of an arc background. 0 deg: right, 90 bottom etc.

参数

- **arc** -- pointer to an arc object
- **end** -- the end angle

`void lv_arc_set_bg_angles(lv_obj_t *arc, uint16_t start, uint16_t end)`

Set the start and end angles of the arc background

参数

- **arc** -- pointer to an arc object
- **start** -- the start angle
- **end** -- the end angle

void lv_arc_set_rotation(*lv_obj_t* *arc, uint16_t rotation)

Set the rotation for the whole arc

参数

- **arc** -- pointer to an arc object
- **rotation** -- rotation angle

void lv_arc_set_mode(*lv_obj_t* *arc, *lv_arc_mode_t* type)

Set the type of arc.

参数

- **arc** -- pointer to arc object
- **mode** -- arc's mode

void lv_arc_set_value(*lv_obj_t* *arc, int16_t value)

Set a new value on the arc

参数

- **arc** -- pointer to a arc object
- **value** -- new value

void lv_arc_set_range(*lv_obj_t* *arc, int16_t min, int16_t max)

Set minimum and the maximum values of a arc

参数

- **arc** -- pointer to the arc object
- **min** -- minimum value
- **max** -- maximum value

void lv_arc_set_change_rate(*lv_obj_t* *arc, uint16_t rate)

Set a change rate to limit the speed how fast the arc should reach the pressed point.

参数

- **arc** -- pointer to a arc object
- **rate** -- the change rate

uint16_t lv_arc_get_angle_start(*lv_obj_t* *obj)

Get the start angle of an arc.

参数 **arc** -- pointer to an arc object

返回 the start angle [0..360]

uint16_t lv_arc_get_angle_end(*lv_obj_t* *obj)

Get the end angle of an arc.

参数 **arc** -- pointer to an arc object

返回 the end angle [0..360]

uint16_t lv_arc_get_bg_angle_start(*lv_obj_t* *obj)

Get the start angle of an arc background.

参数 **arc** -- pointer to an arc object

返回 the start angle [0..360]

```
uint16_t lv_arc_get_bg_angle_end(lv_obj_t *obj)
Get the end angle of an arc background.
```

参数 **arc** -- pointer to an arc object

返回 the end angle [0..360]

```
int16_t lv_arc_get_value(const lv_obj_t *obj)
Get the value of a arc
```

参数 **arc** -- pointer to a arc object

返回 the value of the arc

```
int16_t lv_arc_get_min_value(const lv_obj_t *obj)
Get the minimum value of a arc
```

参数 **arc** -- pointer to a arc object

返回 the minimum value of the arc

```
int16_t lv_arc_get_max_value(const lv_obj_t *obj)
Get the maximum value of a arc
```

参数 **arc** -- pointer to a arc object

返回 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.
```

参数 **arc** -- pointer to a arc object

返回 arc's mode

Variables

```
const lv_obj_class_t lv_arc_class
```

```
struct lv_arc_t
```

Public Members

```
lv_obj_t obj
uint16_t rotation
uint16_t indic_angle_start
uint16_t indic_angle_end
uint16_t bg_angle_start
uint16_t bg_angle_end
int16_t value
int16_t min_value
int16_t max_value
uint16_t dragging
```

```

uint16_t type
uint16_t min_close
uint16_t chg_rate
uint32_t last_tick
int16_t last_angle

```

6.2.2 Bar (进度条) (lv_bar)

Overview (概述)

The bar object has a background and an indicator on it. The width of the indicator is set according to the current value of the bar.

Vertical bars can be created if the width of the object is smaller than its height.

Not only the end, but also the start value of the bar can be set, which changes the start position of the indicator.

进度条对象有一个背景和一个指示器。指标的宽度根据柱线的当前值设置。

如果对象的宽度小于其高度，则可以创建垂直条。

不仅可以设置结束，还可以设置柱线的起始值，从而改变指标的开始位置。

Parts and Styles (零件和样式)

- **LV_PART_MAIN** The background of the bar and it uses the typical background style properties. Adding padding makes the indicator smaller or larger. The `anim_time` style property sets the animation time if the values set with `LV_ANIM_ON`.
- **LV_PART_INDICATOR** The indicator itself; also also uses all the typical background properties.
- **LV_PART_MAIN** 进度条的背景，它使用典型的背景样式属性。添加填充会使指示器变小或变大。如果值设置为“`LV_ANIM_ON`”，则“`anim_time`”样式属性设置动画时间。
- **LV_PART_INDICATOR** 指标本身；还使用所有典型的背景属性。
- **LV_PART_MAIN** 进度条的背景，它使用的是样式属性的背景。补充补充。如果可以变小或变大值设置为“`LV_ANIM_ON`”，则“`anim_time`”样式属性设置动画时间。
- **LV_PART_INDICATOR** 指标本身；还使用所有典型的属性背景。

Usage (用法)

Value and range (值和范围)

A new value can be set by `lv_bar_set_value(bar, new_value, LV_ANIM_ON/OFF)`. The value is interpreted in a range (minimum and maximum values) which can be modified with `lv_bar_set_range(bar, min, max)`. The default range is 1..100.

The new value in `lv_bar_set_value` can be set with or without an animation depending on the last parameter (`LV_ANIM_ON/OFF`).

可以通过 `lv_bar_set_value(bar, new_value, LV_ANIM_ON/OFF)` 设置新值。该值在可以用 `lv_bar_set_range(bar, min, max)` 修改的范围（最小值和最大值）中解释。默认范围是 1..100。

`lv_bar_set_value` 中的新值可以根据最后一个参数 (`LV_ANIM_ON/OFF`) 设置有或没有动画。

Modes (模式)

The bar can be one the following modes:

- `LV_BAR_MODE_NORMAL` A normal bar as described above
- `LV_BAR_SYMMETRICAL` Draw the indicator from the zero value to current value. Requires a negative minimum range and positive maximum range.
- `LV_BAR_RANGE` Allows setting the start value too by `lv_bar_set_start_value(bar, new_value, LV_ANIM_ON/OFF)`. The start value always has to be smaller than the end value.

进度条可以是以下模式之一：

- `LV_BAR_MODE_NORMAL` 如上所述的普通条
- `LV_BAR_SYMMETRICAL` 从零值到当前值绘制指标。需要负的最小范围和正的最大范围。
- `LV_BAR_RANGE` 也允许通过 `lv_bar_set_start_value(bar, new_value, LV_ANIM_ON/OFF)` 设置起始值。起始值必须始终小于结束值。
-

Events (事件)

- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` are sent for the following parts:
 - `LV_BAR_DRAW_PART_INDICATOR` The indicator of the bar
 - * `part: LV_PART_INDICATOR`
 - * `draw_area: area of the indicator`
 - * `rect_dsc`

See the events of the [Base object](#) too.

Learn more about [Events](#).

- 为以下部分发送 `LV_EVENT_DRAW_PART_BEGIN` 和 `LV_EVENT_DRAW_PART_END`:
 - `LV_BAR_DRAW_PART_INDICATOR` 柱线指标
 - * 部分: `LV_PART_INDICATOR`
 - * `draw_area: 指标的区域 -rect_dsc`

参见[Base object](#) 的事件。

详细了解[事件](#)。

Keys (按键)

No *Keys* are processed by the object type.

Learn more about *Keys*.

对象类型不处理 *Keys*。

了解有关*Keys* 的更多信息。

Example

C

Simple Bar

```
#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_scr_act());
    lv_obj_set_size(bar1, 200, 20);
    lv_obj_center(bar1);
    lv_bar_set_value(bar1, 70, LV_ANIM_OFF);
}

#endif
```

```
bar1 = lv.bar(lv.scr_act())
bar1.set_size(200, 30)
bar1.align(None, lv.ALIGN.CENTER, 0, 0)
bar1.set_anim_time(1000)
bar1.set_value(100, lv.ANIM.ON)
```

Styling a bar

```
#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);
```

(下页继续)

(续上页)

```

lv_style_set_anim_time(&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_scr_act());
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

```

Error encountered **while** trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/bar/lv_example_bar_2.py

Temperature meter

```

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

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

```

(下页继续)

(续上页)

```

lv_anim_set_time(&a, 3000);
lv_anim_set_playback_time(&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

```

Error encountered **while** trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/bar/lv_example_bar_3.py

Stripe pattern and range value

```

#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_IMG_DECLARE(img_skew_strip);
    static lv_style_t style_indic;

    lv_style_init(&style_indic);
    lv_style_set_bg_img_src(&style_indic, &img_skew_strip);
    lv_style_set_bg_img_tiled(&style_indic, true);
    lv_style_set_bg_img_opa(&style_indic, LV_OPA_30);

    lv_obj_t * bar = lv_bar_create(lv_scr_act());
    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

```

Error encountered **while** trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/bar/lv_example_bar_4.py

Bar with RTL and RTL base direction

```
#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_scr_act());
    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_scr_act());
    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_scr_act());
    lv_obj_set_base_dir(bar_rtl, LV_BIDI_DIR_RTL);
    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_scr_act());
    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
```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/bar/lv_example_bar_5.py

Custom drawr to show the current value

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

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

static void event_cb(lv_event_t * e)
{
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
    if(dsc->part != LV_PART_INDICATOR) return;

    lv_obj_t * obj= lv_event_get_target(e);
```

(下页继续)

(续上页)

```

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", lv_bar_get_value(obj));

lv_point_t txt_size;
lv_txt_get_size(&txt_size, buf, label_dsc.font, label_dsc.letter_space, label_dsc.
    ↪line_space, LV_COORD_MAX, label_dsc.flag);

lv_area_t txt_area;
/*If the indicator is long enough put the text inside on the right*/
if(lv_area_get_width(dsc->draw_area) > txt_size.x + 20) {
    txt_area.x2 = dsc->draw_area->x2 - 5;
    txt_area.x1 = txt_area.x2 - txt_size.x + 1;
    label_dsc.color = lv_color_white();
}
/*If the indicator is still short put the text out of it on the right*/
else {
    txt_area.x1 = dsc->draw_area->x2 + 5;
    txt_area.x2 = txt_area.x1 + txt_size.x - 1;
    label_dsc.color = lv_color_black();
}

txt_area.y1 = dsc->draw_area->y1 + (lv_area_get_height(dsc->draw_area) - txt_size.
    ↪y) / 2;
txt_area.y2 = txt_area.y1 + txt_size.y - 1;

lv_draw_label(&txt_area, dsc->clip_area, &label_dsc, buf, NULL);
}

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

    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_time(&a, 2000);
    lv_anim_set_playback_time(&a, 2000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

}
#endif

```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/widgets/bar/lv_example_bar_6.py
```

MicroPython

No examples yet.

API

Typedefs

`typedef uint8_t lv_bar_mode_t`

Enums

`enum [anonymous]`

Values:

- enumerator `LV_BAR_MODE_NORMAL`
- enumerator `LV_BAR_MODE_SYMMETRICAL`
- enumerator `LV_BAR_MODE_RANGE`

Functions

`lv_obj_t *lv_bar_create(lv_obj_t *parent)`

Create a bar objects

参数 `parent` -- pointer to an object, it will be the parent of the new bar

返回 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

参数

- `bar` -- 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

参数

- `obj` -- pointer to a bar object
- `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

参数

- **obj** -- pointer to the bar object
- **min** -- minimum value
- **max** -- maximum value

void lv_bar_set_mode(*lv_obj_t* *obj, *lv_bar_mode_t* mode)
Set the type of bar.

参数

- **obj** -- pointer to bar object
- **mode** -- bar type from ::lv_bar_mode_t

int32_t lv_bar_get_value(const *lv_obj_t* *obj)
Get the value of a bar

参数 **obj** -- pointer to a bar object**返回** the value of the bar

int32_t lv_bar_get_start_value(const *lv_obj_t* *obj)
Get the start value of a bar

参数 **obj** -- pointer to a bar object**返回** 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

参数 **obj** -- pointer to a bar object**返回** 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

参数 **obj** -- pointer to a bar object**返回** the maximum value of the bar

***lv_bar_mode_t* lv_bar_get_mode(*lv_obj_t* *obj)**
Get the type of bar.

参数 **obj** -- pointer to bar object**返回** bar type from ::lv_bar_mode_t

Variables

```
const lv_obj_class_t lv_bar_class
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

lv_bar_anim_t **cur_value_anim**

lv_bar_anim_t **start_value_anim**

lv_bar_mode_t **mode**
Type of bar

6.2.3 Button (按钮) (lv_btn)

Overview

Buttons have no new features compared to the *Base object*. They are useful for semantic purposes and have slightly different default settings.

Buttons, by default, differ from Base object in the following ways:

- Not scrollable
- Added to the default group
- Default height and width set to `LV_SIZE_CONTENT`

与*Base object*相比，按钮没有新功能。它们可用于语义目的，并且默认设置略有不同。

默认情况下，按钮在以下方面与 Base 对象不同：

- 不可滚动
- 添加到默认组
- 默认高度和宽度设置为 `LV_SIZE_CONTENT`

Parts and Styles (零件和样式)

- `LV_PART_MAIN` The background of the button. Uses the typical background style properties.

-`LV_PART_MAIN` 按钮的背景。使用典型的背景样式属性。

Usage (用法)

There are no new features compared to *Base object*.

与基本对象相比，没有新功能。

Events (事件)

- `LV_EVENT_VALUE_CHANGED` when the `LV_OBJ_FLAG_CHECKABLE` flag is enabled and the object is clicked. The event happens on transition to/from the checked state.

Learn more about *Events*.

- 启用“`LV_OBJ_FLAG_CHECKABLE`”标志并单击对象时的“`LV_EVENT_VALUE_CHANGED`”。该事件发生在转换到/从检查状态的转换。

详细了解事件。

Keys (按键)

Note that the state of `LV_KEY_ENTER` is translated to `LV_EVENT_PRESSED/PRESSING/RELEASED` etc.

See the events of the [Base object](#) too.

Learn more about [Keys](#).

请注意，“`LV_KEY_ENTER`” 的状态被转换为 “`LV_EVENT_PRESSED/PRESSING/RELEASED`” 等。

参见[Base object](#) 的事件。

了解有关[Keys](#) 的更多信息。

Example

C

Simple Buttons

```
#include "../../lv_examples.h"
#if LV_USE_BTN && 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_btn_1(void)
{
    lv_obj_t * label;

    lv_obj_t * btn1 = lv_btn_create(lv_scr_act());
    lv_obj_add_event_cb(btn1, event_handler, LV_EVENT_ALL, NULL);
    lv_obj_align(btn1, LV_ALIGN_CENTER, 0, -40);

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

    lv_obj_t * btn2 = lv_btn_create(lv_scr_act());
    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);
}
```

(下页继续)

(续上页)

```
}
```

```
#endif
```

```
def event_handler(obj, event):
    if event == lv.EVENT.CLICKED:
        print("Clicked")

btn1 = lv.btn(lv.scr_act())
btn1.set_event_cb(event_handler)
btn1.align(None, lv.ALIGN.CENTER, 0, -40)

label = lv.label(btn1)
label.set_text("Button")

btn2 = lv.btn(lv.scr_act())
# callback can be lambda:
btn2.set_event_cb(lambda obj, event: print("Toggled") if event == lv.EVENT.VALUE_CHANGED else None)
btn2.align(None, lv.ALIGN.CENTER, 0, 40)
btn2.set_toggle(True)
btn2.toggle()
btn2.set_fit2(lv.FIT.NONE, lv.FIT.TIGHT)

label = lv.label(btn2)
label.set_text("Toggled")
```

Styling buttons

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

/***
 * Style a button from scratch
 */
void lv_example_btn_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_darker(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_ofs_y(&style, 8);
```

(下页继续)

(续上页)

```

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_ofs_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 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_btn_create(lv_scr_act());
lv_obj_remove_style_all(btn1);                                /*Remove the style coming
→ from the theme*/
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

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets(btn/lv_example_btn_2.py

Gummy button

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

/***
 * Create a style transition on a button to act like a gum when clicked
 */
void lv_example_btn_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_btn_create(lv_scr_act());
    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
```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets btn/lv_example_btn_3.py

MicroPython

No examples yet.

API

Functions

`lv_obj_t *lv_btn_create(lv_obj_t *parent)`

Create a button object

参数 **parent** -- pointer to an object, it will be the parent of the new button

返回 pointer to the created button

Variables

const lv_obj_class_t **lv_btn_class**

struct **lv_btn_t**

Public Members

`lv_obj_t obj`

6.2.4 Button matrix (按钮矩阵) (`lv_btndmatrix`)

Overview (概述)

The Button Matrix object is a lightweight way to display multiple buttons in rows and columns. Lightweight because the buttons are not actually created but just virtually drawn on the fly. This way, one button use only eight extra bytes of memory instead of the ~100-150 bytes a normal `Button` object plus the 100 or so bytes for the the `Label` object.

The Button matrix is added to the default group (if one is set). Besides the Button matrix is an editable object to allow selecting and clicking the buttons with encoder navigation too.

Button Matrix 对象是一种在行和列中显示多个按钮的轻量级方法。轻量级，因为按钮不是实际创建的，而是实时绘制的。这样，一个按钮仅使用 8 个额外字节的内存，而不是普通 `Button` 对象的 ~100-150 字节加上 [Label]/(widgets/ 核心/标签) 对象。

按钮矩阵被添加到默认组（如果设置了）。此外，按钮矩阵是一个可编辑对象，允许选择和单击带有编码器导航的按钮。

Parts and Styles (零件和样式)

- LV_PART_MAIN The background of the button matrix, uses the typical background style properties. `pad_row` and `pad_column` sets the space between the buttons.
- LV_PART_ITEMS The buttons all use the text and typical background style properties except translations and transformations.
- LV_PART_MAIN 按钮矩阵的背景，使用典型的背景样式属性。`pad_row` 和 `pad_column` 设置按钮之间的空间。
- LV_PART_ITEMS 除了翻译和转换之外，按钮都使用文本和典型的背景样式属性。

Usage (用法)

Button's text (按钮的文字)

There is a text on each button. To specify them a descriptor string array, called *map*, needs to be used. The map can be set with `lv_btnmatrix_set_map(btnm, my_map)`. The declaration of a map should look like `const char * map[] = {"btn1", "btn2", "btn3", NULL}`. Note that the last element has to be either `NULL` or an empty string ("")!

Use "\n" in the map to insert a **line break**. E.g. `{"btn1", "btn2", "\n", "btn3", ""}`. Each line's buttons have their width calculated automatically. So in the example the first row will have 2 buttons each with 50% width and a second row with 1 button having 100% width.

每个按钮上都可以有文字。要指定它们，需要使用称为 *map* 的描述符字符串数组。地图可以用 `lv_btnmatrix_set_map(btnm, my_map)` 设置。映射的声明应该类似于 `const char * map[] = {"btn1", "btn2", "btn3", NULL}`。请注意，最后一个元素必须是 `NULL` 或空字符串("")！

在地图中使用"\n"插入换行符。例如。`{"btn1", "btn2", "\n", "btn3", ""}`。每行按钮的宽度都会自动计算。因此，在示例中，第一行将有 2 个按钮，每个按钮的宽度为 50%，第二行将有 1 个按钮的宽度为 100%。

Control buttons (控制按钮)

The buttons' width can be set relative to the other button in the same row with `lv_btnmatrix_set_btn_width(btnm, btn_id, width)` E.g. in a line with two buttons: `btnA, width = 1` and `btnB, width = 2`, `btnA` will have 33 % width and `btnB` will have 66 % width. It's similar to how the **flex-grow** property works in CSS. The width must be in the [1..7] range and the default width is 1.

In addition to the width, each button can be customized with the following parameters:

- LV_BTNMATRIX_CTRL_HIDDEN Makes a button hidden (hidden buttons still take up space in the layout, they are just not visible or clickable)
- LV_BTNMATRIX_CTRL_NO_REPEAT Disable repeating when the button is long pressed
- LV_BTNMATRIX_CTRL_DISABLED Makes a button disabled Like `LV_STATE_DISABLED` on normal objects
- LV_BTNMATRIX_CTRL_CHECKABLE Enable toggling of a button. I.e. `LV_STATE_CHECKED` will be added/removed as the button is clicked
- LV_BTNMATRIX_CTRL_CHECKED Make the button checked. It will use the `LV_STATE_CHECKED` styles.
- LV_BTNMATRIX_CTRL_CLICK_TRIG Enabled: send `LV_EVENT_VALUE_CHANGE` on CLICK, Disabled: send `LV_EVENT_VALUE_CHANGE` on PRESS*/
- LV_BTNMATRIX_CTRL_RECOLOR Enable recoloring of button texts with #. E.g. "It's #ff0000 red#"

- LV_BTNMATRIX_CTRL_CUSTOM_1 Custom free to use flag
- LV_BTNMATRIX_CTRL_CUSTOM_2 Custom free to use flag

By default all flags are disabled.

可以使用 `lv_btnmatrix_set_btn_width(btnm, btn_id, width)` 相对于同一行中的另一个按钮设置按钮的宽度例如。在一行中有两个按钮: `btnA, width = 1` 和 `btnB, width = 2`, `btnA` 将有 33% 的宽度, `btnB` 将有 66% 的宽度。它类似于 `flex-grow` 属性在 CSS 中的工作方式。宽度必须在 [1..7] 范围内, 默认宽度为 1。

除了宽度之外, 每个按钮还可以使用以下参数进行自定义:

- LV_BTNMATRIX_CTRL_HIDDEN 隐藏按钮 (隐藏的按钮仍然占用布局中的空间, 它们只是不可见或不可点击)
- LV_BTNMATRIX_CTRL_NO_REPEAT 长按按钮时禁用重复
- LV_BTNMATRIX_CTRL_DISABLED 使按钮被禁用, 就像普通对象上的 `LV_STATE_DISABLED`
- LV_BTNMATRIX_CTRL_CHECKABLE 启用按钮切换。IE。`LV_STATE_CHECKED` 将在按钮被点击时添加/删除
- LV_BTNMATRIX_CTRL_CHECKED 检查按钮。它将使用 `LV_STATE_CHECKED` 样式。
- LV_BTNMATRIX_CTRL_CLICK_TRIG 启用: 在点击时发送 `LV_EVENT_VALUE_CHANGE`, 禁用: 在按下时发送 `LV_EVENT_VALUE_CHANGE*/`
- LV_BTNMATRIX_CTRL_RECOLOR 使用 # 启用按钮文本的重新着色。例如。“它是 #ff0000 红色 #”
- LV_BTNMATRIX_CTRL_CUSTOM_1 自定义免费使用标志
- LV_BTNMATRIX_CTRL_CUSTOM_2 自定义免费使用标志

默认情况下, 所有标志都被禁用。

To set or clear a button's control attribute, use `lv_btnmatrix_set_btn_ctrl(btnm, btn_id, LV_BTNM_CTRL_...)` and `lv_btnmatrix_clear_btn_ctrl(btnm, btn_id, LV_BTNMATRIX_CTRL_...)` respectively. More `LV_BTNM_CTRL_...` values can be OR-ed

To set/clear the same control attribute for all buttons of a button matrix, use `lv_btnmatrix_set_btn_ctrl_all(btnm, btn_id, LV_BTNM_CTRL_...)` and `lv_btnmatrix_clear_btn_ctrl_all(btnm, btn_id, LV_BTNMATRIX_CTRL_...)`.

The set a control map for a button matrix (similarly to the map for the text), use `lv_btnmatrix_set_ctrl_map(btnm, ctrl_map)`. An element of `ctrl_map` should look like `ctrl_map[0] = width | LV_BTNM_CTRL_NO_REPEAT | LV_BTNM_CTRL_CHECKABLE`. The number of elements should be equal to the number of buttons (excluding newlines characters).

要设置或清除按钮的控件属性, 请使用 `lv_btnmatrix_set_btn_ctrl(btnm, btn_id, LV_BTNM_CTRL_...)` 和 `lv_btnmatrix_clear_btn_ctrl(btnm, btn_id, LV_BTNMATRIX_CTRL_...)` 分别。更多 `LV_BTNM_CTRL_...` 值可以被 OR-ed

要为按钮矩阵的所有按钮设置/清除相同的控件属性, 请使用 `lv_btnmatrix_set_btn_ctrl_all(btnm, btn_id, LV_BTNM_CTRL_...)` 和 `lv_btnmatrix_clear_btn_ctrl_all(btnm, btn_id, LV_BTNMATRIX_CTRL_...)`。

设置按钮矩阵的控制图 (类似于文本的图) , 使用 `lv_btnmatrix_set_ctrl_map(btnm, ctrl_map)`。`ctrl_map` 的元素应该看起来像 `ctrl_map[0] = width | LV_BTNM_CTRL_NO_REPEAT | LV_BTNM_CTRL_CHECKABLE`。元素的数量应该等于按钮的数量 (不包括换行符)。

One check (一次检查)

The "One check" feature can be enabled with `lv_btnmatrix_set_one_check(btnm, true)` to allow only one button to be checked at a time.

可以使用 `lv_btnmatrix_set_one_check(btnm, true)` 启用“一次检查”功能，以允许一次只检查一个按钮。

Events (事件)

- `LV_EVENT_VALUE_CHANGED` Sent when a button is pressed/released or repeated after long press. The event parameter is set to the ID of the pressed/released button.
- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` are sent for the following types:
 - `LV_BTNMATRIX_DRAW_PART_BTN` The individual buttons.
 - * `part`: `LV_PART_ITEMS`
 - * `id`: index of the button being drawn
 - * `draw_area`: the area of the button
 - * `rect_dsc`
- `LV_EVENT_VALUE_CHANGED` 按下/释放按钮或长按后重复时发送。事件参数设置为按下/释放按钮的 ID。
- 为以下类型发送 `LV_EVENT_DRAW_PART_BEGIN` 和 `LV_EVENT_DRAW_PART_END`:
 - `LV_BTNMATRIX_DRAW_PART_BTN` 单个按钮。
 - * 部分: `LV_PART_ITEMS`
 - * `id`: 正在绘制的按钮的索引
 - * `draw_area`: 按钮的区域
 - * `rect_dsc`

See the events of the [Base object](#) too.

`lv_btnmatrix_get_selected_btn(btnm)` returns the index of the most recently released or focused button or `LV_BTNMATRIX_BTN_NONE` if no such button.

`lv_btnmatrix_get_btn_text(btnm, btn_id)` returns a pointer to the text of `btn_id`th button.

Learn more about [Events](#).

参见[Base object](#) 的事件。

`lv_btnmatrix_get_selected_btn(btnm)` 返回最近释放或聚焦的按钮的索引，如果没有这样的按钮，则返回 `LV_BTNMATRIX_BTN_NONE`。

`lv_btnmatrix_get_btn_text(btnm, btn_id)` 返回指向 `btn_id`th 按钮文本的指针。

详细了解[事件](#)。

Keys (按键)

- LV_KEY_RIGHT/UP/LEFT/RIGHT To navigate among the buttons to select one
- LV_KEY_ENTER To press/release the selected button

Learn more about [Keys](#).

- LV_KEY_RIGHT/UP/LEFT/RIGHT 在按钮之间导航以选择一个
- LV_KEY_ENTER 按下/释放所选按钮

了解有关[Keys](#) 的更多信息。

Example

C

Simple Button matrix

```
#include "../../lv_examples.h"
#if LV_USE_BTNMATRIX && 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(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        uint32_t id = lv_btnmatrix_get_selected_btn(obj);
        const char * txt = lv_btnmatrix_get_btn_text(obj, id);

        LV_LOG_USER("%s was pressed\n", txt);
    }
}

static const char * btnm_map[] = {"1", "2", "3", "4", "5", "\n",
                                 "6", "7", "8", "9", "0", "\n",
                                 "Action1", "Action2", ""};

void lv_example_btnmatrix_1(void)
{
    lv_obj_t * btnm1 = lv_btnmatrix_create(lv_scr_act());
    lv_btnmatrix_set_map(btnm1, btnm_map);
    lv_btnmatrix_set_btn_width(btnm1, 10, 2);           /*Make "Action1" twice as wide as "Action2"*/
    lv_btnmatrix_set_btn_ctrl(btnm1, 10, LV_BTNMATRIX_CTRL_CHECKABLE);
    lv_btnmatrix_set_btn_ctrl(btnm1, 11, LV_BTNMATRIX_CTRL_CHECKED);
    lv_obj_align(btnm1, LV_ALIGN_CENTER, 0, 0);
    lv_obj_add_event_cb(btnm1, event_handler, LV_EVENT_ALL, NULL);
}

#endif
```

```
def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
```

(下页继续)

(续上页)

```

txt = obj.get_active_btn_text()
print("%s was pressed" % txt)

btm_map = ["1", "2", "3", "4", "5", "\n",
           "6", "7", "8", "9", "0", "\n",
           "Action1", "Action2", ""]

btm1 = lv.btm(lv.scr_act())
btm1.set_map(btm_map)
btm1.set_btn_width(10, 2)          # Make "Action1" twice as wide as "Action2"
btm1.align(None, lv.ALIGN.CENTER, 0, 0)
btm1.set_event_cb(event_handler)

```

Custom buttons

```

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

static void event_cb(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_DRAW_PART_BEGIN) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);

        /*Change the draw descriptor the 2nd button*/
        if(dsc->id == 1) {
            dsc->rect_dsc->radius = 0;
            if(lv_btnmatrix_get_selected_btn(obj) == dsc->id)  dsc->rect_dsc->bg_
            ↵color = lv_palette_darken(LV_PALETTE_GREY, 3);
            else dsc->rect_dsc->bg_color = lv_palette_main(LV_PALETTE_BLUE);

            dsc->rect_dsc->shadow_width = 6;
            dsc->rect_dsc->shadow_ofs_x = 3;
            dsc->rect_dsc->shadow_ofs_y = 3;
            dsc->label_dsc->color = lv_color_white();
        }
        /*Change the draw descriptor the 3rd button*/
        else if(dsc->id == 2) {
            dsc->rect_dsc->radius = LV_RADIUS_CIRCLE;
            if(lv_btnmatrix_get_selected_btn(obj) == dsc->id)  dsc->rect_dsc->bg_
            ↵color = lv_palette_darken(LV_PALETTE_RED, 3);
            else dsc->rect_dsc->bg_color = lv_palette_main(LV_PALETTE_RED);

            dsc->label_dsc->color = lv_color_white();
        }
        else if(dsc->id == 3) {
            dsc->label_dsc->opa = LV_OPA_TRANSP; /*Hide the text if any*/
        }
    }
    if(code == LV_EVENT_DRAW_PART_END) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);

```

(下页继续)

(续上页)

```

/*Add custom content to the 4th button when the button itself was drawn*/
if(dsc->id == 3) {
    LV_IMG_DECLARE(img_star);
    lv_img_header_t header;
    lv_res_t res = lv_img_decoder_get_info(&img_star, &header);
    if(res != LV_RES_OK) return;

    lv_area_t a;
    a.x1 = dsc->draw_area->x1 + (lv_area_get_width(dsc->draw_area) - header.
    ↵w) / 2;
    a.x2 = a.x1 + header.w - 1;
    a.y1 = dsc->draw_area->y1 + (lv_area_get_height(dsc->draw_area) - header.
    ↵h) / 2;
    a.y2 = a.y1 + header.h - 1;

    lv_draw_img_dsc_t img_draw_dsc;
    lv_draw_img_dsc_init(&img_draw_dsc);
    img_draw_dsc.recolor = lv_color_black();
    if(lv_btnmatrix_get_selected_btn(obj) == dsc->id) img_draw_dsc.recolor_
    ↵opa = LV_OPA_30;

    lv_draw_img(&a, dsc->clip_area, &img_star, &img_draw_dsc);
}
}

/***
 * Add custom drawer to the button matrix to customize buttons one by one
 */
void lv_example_btnmatrix_2(void)
{
    lv_obj_t * btm = lv_btnmatrix_create(lv_scr_act());
    lv_obj_add_event_cb(btm, event_cb, LV_EVENT_ALL, NULL);
    lv_obj_center(btm);
}

#endif

```

Error encountered **while** trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/btnmatrix/lv_example_btnmatrix_2.py

Pagination

```

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

static void event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    uint32_t id = lv_btnmatrix_get_selected_btn(obj);
    bool prev = id == 0 ? true : false;
    bool next = id == 6 ? true : false;
    if(prev || next) {

```

(下页继续)

(续上页)

```

/*Find the checked button*/
uint32_t i;
for(i = 1; i < 7; i++) {
    if(lv_btnmatrix_has_btn_ctrl(obj, i, LV_BTNMATRIX_CTRL_CHECKED)) break;
}

if(prev && i > 1) i--;
else if(next && i < 5) i++;

lv_btnmatrix_set_btn_ctrl(obj, i, LV_BTNMATRIX_CTRL_CHECKED);
}

/***
 * Make a button group (pagination)
 */
void lv_example_btnmatrix_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 * bnm = lv_btnmatrix_create(lv_scr_act());
    lv_btnmatrix_set_map(bnm, map);
    lv_obj_add_style(bnm, &style_bg, 0);
    lv_obj_add_style(bnm, &style_btn, LV_PART_ITEMS);
    lv_obj_add_event_cb(bnm, event_cb, LV_EVENT_VALUE_CHANGED, NULL);
    lv_obj_set_size(bnm, 225, 35);

    /*Allow selecting on one number at time*/
    lv_btnmatrix_set_btn_ctrl_all(bnm, LV_BTNMATRIX_CTRL_CHECKABLE);
    lv_btnmatrix_clear_btn_ctrl(bnm, 0, LV_BTNMATRIX_CTRL_CHECKABLE);
    lv_btnmatrix_clear_btn_ctrl(bnm, 6, LV_BTNMATRIX_CTRL_CHECKABLE);

    lv_btnmatrix_set_one_checked(bnm, true);
    lv_btnmatrix_set_btn_ctrl(bnm, 1, LV_BTNMATRIX_CTRL_CHECKED);

    lv_obj_center(bnm);
}

```

(下页继续)

(续上页)

```
}
```

```
#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/widgets/btnmatrix/lv_example_btnmatrix_3.py
```

MicroPython

No examples yet.

API

Typedefs

```
typedef uint16_t lv_btnmatrix_ctrl_t
typedef bool (*lv_btnmatrix_btn_draw_cb_t)(lv_obj_t *btnm, uint32_t btn_id, const lv_area_t *draw_area,
const lv_area_t *clip_area)
```

Enums

enum [**anonymous**]

Type to store button control bits (disabled, hidden etc.) The first 3 bits are used to store the width

Values:

enumerator **LV_BTNMATRIX_WIDTH**

Reserved to store the size units

enumerator **LV_BTNMATRIX_CTRL_HIDDEN**

Button hidden

enumerator **LV_BTNMATRIX_CTRL_NO_REPEAT**

Do not repeat press this button.

enumerator **LV_BTNMATRIX_CTRL_DISABLED**

Disable this button.

enumerator **LV_BTNMATRIX_CTRL_CHECKABLE**

The button can be toggled.

enumerator **LV_BTNMATRIX_CTRL_CHECKED**

Button is currently toggled (e.g. checked).

enumerator **LV_BTNMATRIX_CTRL_CLICK_TRIG**

1: Send LV_EVENT_VALUE_CHANGE on CLICK, 0: Send LV_EVENT_VALUE_CHANGE on PRESS

enumerator **LV_BTNMATRIX_CTRL_RECOLOR**

Enable text recoloring with #color

enumerator **_LV_BTNMATRIX_CTRL_RESERVED**

Reserved for later use

enumerator **LV_BTNMATRIX_CTRL_CUSTOM_1**

Custom free to use flag

enumerator **LV_BTNMATRIX_CTRL_CUSTOM_2**

Custom free to use flag

Functions

LV_EXPORT_CONST_INT(LV_BTNMATRIX_BTN_NONE)

lv_obj_t ***lv_btnmatrix_create**(*lv_obj_t* *parent)

Create a button matrix objects

参数 **parent** -- pointer to an object, it will be the parent of the new button matrix

返回 pointer to the created button matrix

void lv_btnmatrix_set_map(*lv_obj_t* *obj, const char *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.

参数

- **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_btnmatrix_set_ctrl_map(*lv_obj_t* *obj, const *lv_btnmatrix_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.

参数

- **obj** -- pointer to a button matrix object
- **ctrl_map** -- pointer to an array of *lv_btn_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_BTNMATRIX_CTRL_NO_REPEAT | LV_BTNMATRIX_CTRL_TGL_ENABLE`

void lv_btnmatrix_set_selected_btn(*lv_obj_t* *obj, uint16_t btn_id)

Set the selected buttons

参数

- **obj** -- pointer to button matrix object
- **btn_id** -- 0 based index of the button to modify. (Not counting new lines)

void lv_btnmatrix_set_btn_ctrl(*lv_obj_t* *obj, uint16_t btn_id, *lv_btnmatrix_ctrl_t* ctrl)

Set the attributes of a button of the button matrix

参数

- **obj** -- pointer to button matrix object
- **btn_id** -- 0 based index of the button to modify. (Not counting new lines)
- **ctrl** -- OR-ed attributs. E.g. `LV_BTNMATRIX_CTRL_NO_REPEAT | LV_BTNMATRIX_CTRL_CHECKABLE`

void lv_btnmatrix_clear_btn_ctrl (const *lv_obj_t* *obj, uint16_t btn_id, *lv_btnmatrix_ctrl_t* ctrl)

Clear the attributes of a button of the button matrix

参数

- **obj** -- pointer to button matrix object
- **btn_id** -- 0 based index of the button to modify. (Not counting new lines)
- **ctrl** -- OR-ed attributs. E.g. `LV_BTNMATRIX_CTRL_NO_REPEAT | LV_BTNMATRIX_CTRL_CHECKABLE`

void lv_btnmatrix_set_btn_ctrl_all (*lv_obj_t* *obj, *lv_btnmatrix_ctrl_t* ctrl)

Set attributes of all buttons of a button matrix

参数

- **obj** -- pointer to a button matrix object
- **ctrl** -- attribute(s) to set from `lv_btnmatrix_ctrl_t`. Values can be ORed.

void lv_btnmatrix_clear_btn_ctrl_all (*lv_obj_t* *obj, *lv_btnmatrix_ctrl_t* ctrl)

Clear the attributes of all buttons of a button matrix

参数

- **obj** -- pointer to a button matrix object
- **ctrl** -- attribute(s) to set from `lv_btnmatrix_ctrl_t`. Values can be ORed.
- **en** -- true: set the attributes; false: clear the attributes

void lv_btnmatrix_set_btn_width (*lv_obj_t* *obj, uint16_t btn_id, uint8_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_btnmatrix_set_ctrl_map` and this method only be used for dynamic changes.

参数

- **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..7]

void lv_btnmatrix_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_BTNMATRIX_CTRL_CHECKABLE` must be enabled on the buttons to be selected using `lv_btnmatrix_set_ctrl()` or `lv_btnmatrix_set_btn_ctrl_all()`.

参数

- **obj** -- pointer to a button matrix object
- **en** -- whether "one check" mode is enabled

const char **lv_btnmatrix_get_map (const *lv_obj_t* *obj)

Get the current map of a button matrix

参数 **obj** -- pointer to a button matrix object

返回 the current map

uint16_t lv_btnmatrix_get_selected_btn(const *lv_obj_t* *obj)

Get the index of the lastly "activated" button by the user (pressed, released, focused etc) Useful in the the event_cb to get the text of the button, check if hidden etc.

参数 **obj** -- pointer to button matrix object

返回 index of the last released button (LV_BTNMATRIX_BTN_NONE: if unset)

const char *lv_btnmatrix_get_btn_text(const *lv_obj_t* *obj, uint16_t btn_id)

Get the button's text

参数

- **obj** -- pointer to button matrix object

- **btn_id** -- the index a button not counting new line characters.

返回 text of btn_index^c button

bool lv_btnmatrix_has_btn_ctrl(*lv_obj_t* *obj, uint16_t btn_id, *lv_btnmatrix_ctrl_t* ctrl)

Get the whether a control value is enabled or disabled for button of a button matrix

参数

- **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)

返回 true: the control attribute is enabled false: disabled

bool lv_btnmatrix_get_one_checked(const *lv_obj_t* *obj)

Tell whether "one check" mode is enabled or not.

参数 **obj** -- Button matrix object

返回 true: "one check" mode is enabled; false: disabled

Variables

const lv_obj_class_t **lv_btnmatrix_class**

struct **lv_btnmatrix_t**

Public Members

lv_obj_t **obj**

const char ****map_p**

lv_area_t ***button_areas**

lv_btnmatrix_ctrl_t ***ctrl_bits**

uint16_t **btn_cnt**

uint16_t **btn_id_sel**

```
uint8_t one_check
```

6.2.5 Canvas (画布) (lv_canvas)

Overview (概述)

A Canvas inherits from [Image](#) where the user can draw anything. Rectangles, texts, images, lines, arcs can be drawn here using lvgl's drawing engine. Additionally "effects" can be applied, such as rotation, zoom and blur.

Canvas 继承自 [Image](#)，用户可以在其中绘制任何内容。矩形、文本、图像、线条、圆弧可以在这里使用 lvgl 的绘图引擎绘制。此外，可以应用“效果”，例如旋转、缩放和模糊。

Parts and Styles (零件和风格)

LV_PART_MAIN Uses the typical rectangle style properties and image style properties.

LV_PART_MAIN 使用典型的矩形样式属性和图像样式属性。

Usage (用法)

Buffer (缓冲区)

The Canvas needs a buffer in which stores the drawn image. To assign a buffer to a Canvas, use `lv_canvas_set_buffer(canvas, buffer, width, height, LV_IMG_CF_...)`. Where buffer is a static buffer (not just a local variable) to hold the image of the canvas. For example, `static lv_color_t buffer[LV_CANVAS_BUF_SIZE_TRUE_COLOR(width, height)]`. `LV_CANVAS_BUF_SIZE_...` macros help to determine the size of the buffer with different color formats.

The canvas supports all the built-in color formats like `LV_IMG_CF_TRUE_COLOR` or `LV_IMG_CF_INDEXED_2BIT`. See the full list in the [Color formats](#) section.

Canvas 需要一个缓冲区来存储绘制的图像。要为 Canvas 分配缓冲区，请使用 `lv_canvas_set_buffer(canvas, buffer, width, height, LV_IMG_CF_...)`。其中 buffer 是一个静态缓冲区（不仅仅是一个局部变量）来保存画布的图像。例如，静态 `lv_color_t` 缓冲区 `[LV_CANVAS_BUF_SIZE_TRUE_COLOR(width, height)]`。`LV_CANVAS_BUF_SIZE_...` 宏有助于确定具有不同颜色格式的缓冲区的大小。

画布支持所有内置颜色格式，如“`LV_IMG_CF_TRUE_COLOR`”或“`LV_IMG_CF_INDEXED_2BIT`”。请参阅 [颜色格式](#) 部分中的完整列表。

Indexed colors (颜色索引)

For `LV_IMG_CF_INDEXED_1/2/4/8` color formats a palette needs to be initialized with `lv_canvas_set_palette(canvas, 3, LV_COLOR_RED)`. It sets pixels with `index=3` to red.

对于“`LV_IMG_CF_INDEXED_1/2/4/8`”颜色格式，调色板需要用 `lv_canvas_set_palette(canvas, 3, LV_COLOR_RED)` 初始化。它将 `index=3` 的像素设置为红色。

Drawing (画画)

To set a pixel on the canvas, use `lv_canvas_set_px(canvas, x, y, LV_COLOR_RED)`. With `LV_IMG_CF_INDEXED_...` or `LV_IMG_CF_ALPHA_...`, the index of the color or the alpha value needs to be passed as color. E.g. `lv_color_t c; c.full = 3;`

`lv_canvas_fill_bg(canvas, LV_COLOR_BLUE, LV_OPA_50)` fills the whole canvas to blue with 50% opacity. Note that if the current color format doesn't support colors (e.g. `LV_IMG_CF_ALPHA_2BIT`) the color will be ignored. Similarly, if opacity is not supported (e.g. `LV_IMG_CF_TRUE_COLOR`) it will be ignored.

An array of pixels can be copied to the canvas with `lv_canvas_copy_buf(canvas, buffer_to_copy, x, y, width, height)`. The color format of the buffer and the canvas need to match.

要在画布上设置像素，请使用 `lv_canvas_set_px(canvas, x, y, LV_COLOR_RED)`。使用 `LV_IMG_CF_INDEXED_...` 或 `LV_IMG_CF_ALPHA_...`，颜色的索引或 alpha 值需要作为颜色传递。例如。`lv_color_t c; c.full = 3;`

`lv_canvas_fill_bg(canvas, LV_COLOR_BLUE, LV_OPA_50)` 将整个画布填充为蓝色，不透明度为 50%。请注意，如果当前颜色格式不支持颜色（例如 `LV_IMG_CF_ALPHA_2BIT`），则颜色将被忽略。同样，如果不支持不透明度（例如 `LV_IMG_CF_TRUE_COLOR`），它将被忽略。

可以使用 `lv_canvas_copy_buf(canvas, buffer_to_copy, x, y, width, height)` 将像素数组复制到画布。缓冲区和画布的颜色格式需要匹配。

To draw something to the canvas use

- `lv_canvas_draw_rect(canvas, x, y, width, height, &draw_dsc)`
- `lv_canvas_draw_text(canvas, x, y, max_width, &draw_dsc, txt)`
- `lv_canvas_draw_img(canvas, x, y, &img_src, &draw_dsc)`
- `lv_canvas_draw_line(canvas, point_array, point_cnt, &draw_dsc)`
- `lv_canvas_draw_polygon(canvas, points_array, point_cnt, &draw_dsc)`
- `lv_canvas_draw_arc(canvas, x, y, radius, start_angle, end_angle, &draw_dsc)`

`draw_dsc` is a `lv_draw_rect/label/img/line/arc_dsc_t` variable which should be first initialized with one of `lv_draw_rect/label/img/line/arc_dsc_init()` and then modified with the desired colors and other values.

The draw function can draw to any color format. For example, it's possible to draw a text to an `LV_IMG_VF_ALPHA_8BIT` canvas and use the result image as a *draw mask* later.

要在画布上绘制一些东西，请使用

- `lv_canvas_draw_rect(canvas, x, y, width, height, &draw_dsc)`
- `lv_canvas_draw_text(canvas, x, y, max_width, &draw_dsc, txt)`
- `lv_canvas_draw_img(canvas, x, y, &img_src, &draw_dsc)`
- `lv_canvas_draw_line(canvas, point_array, point_cnt, &draw_dsc)`
- `lv_canvas_draw_polygon(canvas, points_array, point_cnt, &draw_dsc)`
- `lv_canvas_draw_arc(canvas, x, y, radius, start_angle, end_angle, &draw_dsc)`

`draw_dsc` 是一个 `lv_draw_rect/label/img/line/arc_dsc_t` 变量，它应该首先使用 `lv_draw_rect/label/img/line/arc_dsc_init()` 中的一个进行初始化，然后使用所需的颜色和其他值进行修改。

`draw` 函数可以绘制成任何颜色格式。例如，可以在“LV_IMG_VF_ALPHA_8BIT”画布上绘制文本，然后将结果图像用作绘制蒙版。

Transformations (变换)

`lv_canvas_transform()` 可以用于旋转和/或缩放图像的图像并将结果存储在画布上。该函数需要以下参数：

- `canvas` 指向一个画布对象的指针，用于存储转换的结果。
- `img` 指向要转换的图像描述符的“img 指针”。也可以是其他画布的图像描述符 (`lv_canvas_get_img()`)。
- `angle` 旋转角度 (0..3600), 0.1 度分辨率
- `zoom` 缩放系数 (256: 无缩放, 512: 双倍尺寸, 128: 半尺寸);
- `offset_x` 偏移 X 来告诉将结果数据放在目标画布上的什么位置
- `offset_y` 偏移 Y 来告诉将结果数据放在目标画布上的什么位置
- `pivot_x` 旋转的枢轴 X。相对于源画布。设置为 `source width / 2` 以围绕中心旋转
- `pivot_y` 旋转轴 Y。相对于源画布。设置为 `source height / 2` 以围绕中心旋转
- `antialias` true: 在转换过程中应用抗锯齿。看起来更好但更慢。

Note that a canvas can't be rotated on itself. You need a source and destination canvas or image.

`lv_canvas_transform()` 可用于旋转和/或缩放图像的图像并将结果存储在画布上。该函数需要以下参数：

- `canvas` 指向一个画布对象的指针，用于存储转换的结果。
- 指向要转换的图像描述符的“img 指针”。也可以是其他画布的图像描述符 (`lv_canvas_get_img()`)。
- `angle` 旋转角度 (0..3600), 0.1 度分辨率
- `zoom` 缩放系数 (256: 无缩放, 512: 双倍尺寸, 128: 半尺寸);
- `offset_x` 偏移 X 来告诉将结果数据放在目标画布上的什么位置
- `offset_y` 偏移 Y 来告诉将结果数据放在目标画布上的什么位置
- `pivot_x` 旋转的枢轴 X。相对于源画布。设置为 `source width / 2` 以围绕中心旋转
- `pivot_y` 旋转轴 Y。相对于源画布。设置为 `source height / 2` 以围绕中心旋转
- `antialias` true: 在转换过程中应用抗锯齿。看起来更好但更慢。

请注意，画布不能自行旋转。您需要一个源和目标画布或图像。

Blur (糊化)

A given area of the canvas can be blurred horizontally with `lv_canvas_blur_hor(canvas, &area, r)` or vertically with `lv_canvas_blur_ver(canvas, &area, r)`. `r` is the radius of the blur (greater value means more intensive blurring). `area` is the area where the blur should be applied (interpreted relative to the canvas).

画布的给定区域可以使用 `lv_canvas_blur_hor(canvas, &area, r)` 进行水平模糊处理，或者使用 `lv_canvas_blur_ver(canvas, &area, r)` 进行垂直模糊处理。`r` 是模糊的半径（值越大意味着毛刺越强）。`area` 是应该应用模糊的区域（相对于画布进行解释）。

Events (事件)

No special events are sent by canvas objects. The same events are sent as for the

See the events of the [Images](#) too.

Learn more about [Events](#).

画布对象不会发送特殊事件。

也可以查看 [Images](#) 的事件。

详细了解 [事件](#)。

Keys (按键)

No *Keys* are processed by the object type.

Learn more about [Keys](#).

对象类型不处理 *Keys*。

了解有关 [Keys](#) 的更多信息。

Example

C

Drawing on the Canvas and rotate

```
#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_HOR;
    rect_dsc.bg_color = lv_palette_main(LV_PALETTE_RED);
```

(下页继续)

(续上页)

```

rect_dsc.bg_grad_color = lv_palette_main(LV_PALETTE_BLUE);
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_ofs_x = 5;
rect_dsc.shadow_ofs_y = 5;

lv_draw_label_dsc_t label_dsc;
lv_draw_label_dsc_init(&label_dsc);
label_dsc.color = lv_palette_main(LV_PALETTE_YELLOW);

static lv_color_t cbuf[LV_CANVAS_BUF_SIZE_TRUE_COLOR(CANVAS_WIDTH, CANVAS_
HEIGHT)];

lv_obj_t * canvas = lv_canvas_create(lv_scr_act());
lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_IMG_CF_TRUE_
COLOR);
lv_obj_center(canvas);
lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);

lv_canvas_draw_rect(canvas, 70, 60, 100, 70, &rect_dsc);

lv_canvas_draw_text(canvas, 40, 20, 100, &label_dsc, "Some text on text canvas");

/*Test the rotation. It requires an other buffer where the original image is_
stored.
*So copy the current image to buffer and rotate it to the canvas*/
static lv_color_t cbuf_tmp[CANVAS_WIDTH * CANVAS_HEIGHT];
memcpy(cbuf_tmp, cbuf, sizeof(cbuf_tmp));
lv_img_dsc_t img;
img.data = (void *)cbuf_tmp;
img.header.cf = LV_IMG_CF_TRUE_COLOR;
img.header.w = CANVAS_WIDTH;
img.header.h = CANVAS_HEIGHT;

lv_canvas_fill_bg(canvas, lv_palette_lighten(LV_PALETTE_GREY, 3), LV_OPA_COVER);
lv_canvas_transform(canvas, &img, 30, LV_IMG_ZOOM_NONE, 0, 0, CANVAS_WIDTH / 2,
CANVAS_HEIGHT / 2, true);
}

#endif

```

```

CANVAS_WIDTH = 200
CANVAS_HEIGHT = 150

style = lv.style_t()
lv.style_copy(style, lv.style_plain)
style.body.main_color = lv.color_make(0xFF, 0, 0)
style.body.grad_color = lv.color_make(0x80, 0, 0)
style.body.radius = 4
style.body.border.width = 2
style.body.border.color = lv.color_make(0xFF, 0xFF, 0xFF)
style.body.shadow.color = lv.color_make(0xFF, 0xFF, 0xFF)
style.body.shadow.width = 4
style.line.width = 2

```

(下页继续)

(续上页)

```

style.line.color = lv.color_make(0,0,0)
style.text.color = lv.color_make(0,0,0xFF)

# CF.TRUE_COLOR requires 4 bytes per pixel
cbuf = bytearray(CANVAS_WIDTH * CANVAS_HEIGHT * 4)

canvas = lv.canvas(lv.scr_act())
canvas.set_buffer(cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, lv.img.CF.TRUE_COLOR)
canvas.align(None, lv.ALIGN.CENTER, 0, 0)
canvas.fill_bg(lv.color_make(0xC0, 0xC0, 0xC0))

canvas.draw_rect(70, 60, 100, 70, style)

canvas.draw_text(40, 20, 100, style, "Some text on text canvas", lv.label.ALIGN.LEFT)

# Test the rotation. It requires an other buffer where the original image is stored.
# So copy the current image to buffer and rotate it to the canvas
img = lv.img_dsc_t()
img.data = cbuf[:]
img.header.cf = lv.img.CF.TRUE_COLOR
img.header.w = CANVAS_WIDTH
img.header.h = CANVAS_HEIGHT

canvas.fill_bg(lv.color_make(0xC0, 0xC0, 0xC0))
canvas.rotate(img, 30, 0, 0, CANVAS_WIDTH // 2, CANVAS_HEIGHT // 2)

```

Transparent Canvas with chroma keying

```

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

#define CANVAS_WIDTH 50
#define CANVAS_HEIGHT 50

/**
 * Create a transparent canvas with Chroma keying and indexed color format (palette).
 */
void lv_example_canvas_2(void)
{
    /*Create a button to better see the transparency*/
    lv_btn_create(lv_scr_act());

    /*Create a buffer for the canvas*/
    static lv_color_t cbuf[LV_CANVAS_BUF_SIZE_INDEXED_1BIT(CANVAS_WIDTH, CANVAS_
    HEIGHT)];

    /*Create a canvas and initialize its the palette*/
    lv_obj_t * canvas = lv_canvas_create(lv_scr_act());
    lv_canvas_set_buffer(canvas, cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, LV_IMG_CF_INDEXED_
    1BIT);
    lv_canvas_set_palette(canvas, 0, LV_COLOR_CHROMA_KEY);
    lv_canvas_set_palette(canvas, 1, lv_palette_main(LV_PALETTE_RED));

    /*Create colors with the indices of the palette*/

```

(下页继续)

(续上页)

```

lv_color_t c0;
lv_color_t c1;

c0.full = 0;
c1.full = 1;

/*Red background (There is no dedicated alpha channel in indexed images so LV_OPA_COVER is ignored)*/
lv_canvas_fill_bg(canvas, c1, LV_OPA_COVER);

/*Create hole on the canvas*/
uint32_t x;
uint32_t y;
for( y = 10; y < 30; y++) {
    for( x = 5; x < 20; x++) {
        lv_canvas_set_px(canvas, x, y, c0);
    }
}

#endif

```

```

# Create a transparent canvas with Chroma keying and indexed color format (palette).

CANVAS_WIDTH = 50
CANVAS_HEIGHT = 50

def bufsize(w, h, bits, indexed=False):
    """this function determines required buffer size
       depending on the color depth"""
    size = (w * bits // 8 + 1) * h
    if indexed:
        # + 4 bytes per palette color
        size += 4 * (2**bits)
    return size

# Create a button to better see the transparency
lv.btn(lv.scr_act())

# Create a buffer for the canvas
cbuf = bytearray(bufsize(CANVAS_WIDTH, CANVAS_HEIGHT, 1, indexed=True))

# Create a canvas and initialize its the palette
canvas = lv.canvas(lv.scr_act())
canvas.set_buffer(cbuf, CANVAS_WIDTH, CANVAS_HEIGHT, lv.img.CF.INDEXED_1BIT)
# transparent color can be defined in lv_conf.h and set to pure green by default
canvas.set_palette(0, lv.color_make(0x00, 0xFF, 0x00))
canvas.set_palette(1, lv.color_make(0xFF, 0x00, 0x00))

# Create colors with the indices of the palette
c0 = lv.color_t()
c1 = lv.color_t()

c0.full = 0
c1.full = 1

```

(下页继续)

(续上页)

```
# Transparent background
canvas.fill_bg(c1)

# Create hole on the canvas
for y in range(10,30):
    for x in range(5, 20):
        canvas.set_px(x, y, c0)
```

MicroPython

No examples yet.

API

Functions

`lv_obj_t *lv_canvas_create(lv_obj_t *parent)`

Create a canvas object

参数 **parent** -- pointer to an object, it will be the parent of the new canvas

返回 pointer to the created canvas

`void lv_canvas_set_buffer(lv_obj_t *canvas, void *buf, lv_coord_t w, lv_coord_t h, lv_img_cf_t cf)`

Set a buffer for the canvas.

参数

- **buf** -- a buffer where the content of the canvas will be. The required size is `(lv_img_color_format_get_px_size(cf) * w) / 8 * h`. It can be allocated with `lv_mem_alloc()` 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
- **canvas** -- pointer to a canvas object
- **w** -- width of the canvas
- **h** -- height of the canvas
- **cf** -- color format. `LV_IMG_CF_...`

`void lv_canvas_set_px(lv_obj_t *canvas, lv_coord_t x, lv_coord_t y, lv_color_t c)`

Set the color of a pixel on the canvas

参数

- **canvas** --
- **x** -- x coordinate of the point to set
- **y** -- x coordinate of the point to set
- **c** -- color of the point

`void lv_canvas_set_palette(lv_obj_t *canvas, uint8_t id, lv_color_t c)`

Set the palette color of a canvas with index format. Valid only for `LV_IMG_CF_INDEXED1/2/4/8`

参数

- **canvas** -- pointer to canvas object

- **id** -- the palette color to set:
 - for LV_IMG_CF_INDEXED1: 0..1
 - for LV_IMG_CF_INDEXED2: 0..3
 - for LV_IMG_CF_INDEXED4: 0..15
 - for LV_IMG_CF_INDEXED8: 0..255
- **c** -- the color to set

`lv_color_t lv_canvas_get_px(lv_obj_t *canvas, lv_coord_t x, lv_coord_t y)`

Get the color of a pixel on the canvas

参数

- **canvas** --
- **x** -- x coordinate of the point to set
- **y** -- x coordinate of the point to set

返回 color of the point

`lv_img_dsc_t *lv_canvas_get_img(lv_obj_t *canvas)`

Get the image of the canvas as a pointer to an `lv_img_dsc_t` variable.

参数 **canvas** -- pointer to a canvas object

返回 pointer to the image descriptor.

`void lv_canvas_copy_buf(lv_obj_t *canvas, const void *to_copy, lv_coord_t x, lv_coord_t y, lv_coord_t w, lv_coord_t h)`

Copy a buffer to the canvas

参数

- **canvas** -- pointer to a canvas object
- **to_copy** -- buffer to copy. The color format has to match with the canvas's buffer color format
 - **x** -- left side of the destination position
 - **y** -- top side of the destination position
 - **w** -- width of the buffer to copy
 - **h** -- height of the buffer to copy

`void lv_canvas_transform(lv_obj_t *canvas, lv_img_dsc_t *img, int16_t angle, uint16_t zoom, lv_coord_t offset_x, lv_coord_t offset_y, int32_t pivot_x, int32_t pivot_y, bool antialias)`

Transform and image and store the result on a canvas.

参数

- **canvas** -- pointer to a canvas object to store the result of the transformation.
- **img** -- pointer to an image descriptor to transform. Can be the image descriptor of an other canvas too (`lv_canvas_get_img()`).
- **angle** -- the angle of rotation (0..3600), 0.1 deg resolution
- **zoom** -- zoom factor (256 no zoom);
- **offset_x** -- offset X to tell where to put the result data on destination canvas
- **offset_y** -- offset Y to tell where to put the result data on destination canvas

- **pivot_x** -- pivot X of rotation. Relative to the source canvas Set to `source width / 2` to rotate around the center
- **pivot_y** -- pivot Y of rotation. Relative to the source canvas Set to `source height / 2` to rotate around the center
- **antialias** -- apply anti-aliasing during the transformation. Looks better but slower.

`void lv_canvas_blur_hor(lv_obj_t *canvas, const lv_area_t *area, uint16_t r)`
Apply horizontal blur on the canvas

参数

- **canvas** -- pointer to a canvas object
- **area** -- the area to blur. If NULL the whole canvas will be blurred.
- **r** -- radius of the blur

`void lv_canvas_blur_ver(lv_obj_t *canvas, const lv_area_t *area, uint16_t r)`
Apply vertical blur on the canvas

参数

- **canvas** -- pointer to a canvas object
- **area** -- the area to blur. If NULL the whole canvas will be blurred.
- **r** -- radius of the blur

`void lv_canvas_fill_bg(lv_obj_t *canvas, lv_color_t color, lv_opa_t opa)`
Fill the canvas with color

参数

- **canvas** -- pointer to a canvas
- **color** -- the background color
- **opa** -- the desired opacity

`void lv_canvas_draw_rect(lv_obj_t *canvas, lv_coord_t x, lv_coord_t y, lv_coord_t w, lv_coord_t h, const lv_draw_rect_dsc_t *draw_dsc)`
Draw a rectangle on the canvas

参数

- **canvas** -- pointer to a canvas object
- **x** -- left coordinate of the rectangle
- **y** -- top coordinate of the rectangle
- **w** -- width of the rectangle
- **h** -- height of the rectangle
- **draw_dsc** -- descriptor of the rectangle

`void lv_canvas_draw_text(lv_obj_t *canvas, lv_coord_t x, lv_coord_t y, lv_coord_t max_w, lv_draw_label_dsc_t *draw_dsc, const char *txt)`

Draw a text on the canvas.

参数

- **canvas** -- pointer to a canvas object
- **x** -- left coordinate of the text

- **y** -- top coordinate of the text
- **max_w** -- max width of the text. The text will be wrapped to fit into this size
- **draw_dsc** -- pointer to a valid label descriptor `lv_draw_label_dsc_t`
- **txt** -- text to display

```
void lv_canvas_draw_img(lv_obj_t*canvas, lv_coord_t x, lv_coord_t y, const void *src, const
                      lv_draw_img_dsc_t *draw_dsc)
```

Draw an image on the canvas

参数

- **canvas** -- pointer to a canvas object
- **x** -- left coordinate of the image
- **y** -- top coordinate of the image
- **src** -- image source. Can be a pointer an `lv_img_dsc_t` variable or a path an image.
- **draw_dsc** -- pointer to a valid label descriptor `lv_draw_img_dsc_t`

```
void lv_canvas_draw_line(lv_obj_t*canvas, const lv_point_t points[], uint32_t point_cnt, const
                       lv_draw_line_dsc_t *draw_dsc)
```

Draw a line on the canvas

参数

- **canvas** -- pointer to a canvas object
- **points** -- point of the line
- **point_cnt** -- number of points
- **draw_dsc** -- pointer to an initialized `lv_draw_line_dsc_t` variable

```
void lv_canvas_draw_polygon(lv_obj_t*canvas, const lv_point_t points[], uint32_t point_cnt, const
                           lv_draw_rect_dsc_t *draw_dsc)
```

Draw a polygon on the canvas

参数

- **canvas** -- pointer to a canvas object
- **points** -- point of the polygon
- **point_cnt** -- number of points
- **draw_dsc** -- pointer to an initialized `lv_draw_rect_dsc_t` variable

```
void lv_canvas_draw_arc(lv_obj_t*canvas, lv_coord_t x, lv_coord_t y, lv_coord_t r, int32_t start_angle,
                       int32_t end_angle, const lv_draw_arc_dsc_t *draw_dsc)
```

Draw an arc on the canvas

参数

- **canvas** -- pointer to a canvas object
- **x** -- origo x of the arc
- **y** -- origo y of the arc
- **r** -- radius of the arc
- **start_angle** -- start angle in degrees
- **end_angle** -- end angle in degrees

- **draw_dsc** -- pointer to an initialized `lv_draw_line_dsc_t` variable

struct `lv_canvas_t`

Public Members

`lv_img_t img`

`lv_img_dsc_t dsc`

6.2.6 Checkbox (复选框) (`lv_checkbox`)

Overview (概述)

The Checkbox object is created from a "tick box" and a label. When the Checkbox is clicked the tick box is toggled.

复选框 (Checkbox) 对象是从“勾选框”和标签创建的。当 Checkbox 被点击时，勾选框被切换。

Parts and Styles (部件和样式)

- **LV_PART_MAIN** The is the background of the Checkbox and it uses the text and all the typical background style properties. `pad_column` adjusts the spacing between the tickbox and the label
- **LV_PART_INDICATOR** The "tick box" is a square that uses all the typical background style properties. By default its size is equal to the height of the main part's font. Padding properties make the tick box larger in the respective directions.

The Checkbox is added to the default group (if it is set).

- **LV_PART_MAIN** 这是复选框的背景，它使用文本和所有典型的背景样式属性。`pad_column` 调整复选框和标签之间的间距
- **LV_PART_INDICATOR** “勾选框”是一个使用所有典型背景样式属性的正方形。默认情况下，它的大小等于主要部分字体的高度。填充属性使复选框在相应方向上变大。

复选框将添加到默认组（如果已设置）。

Usage (用法)

Text () 文本

The text can be modified with the `lv_checkbox_set_text(cb, "New text")` function and will be dynamically allocated.

To set a static text, use `lv_checkbox_set_static_text(cb, txt)`. This way, only a pointer to `txt` will be stored. The text then shouldn't be deallocated while the checkbox exists.

文本可以使用 `lv_checkbox_set_text(cb, "New text")` 函数进行修改，并将被动态分配。

要设置静态文本，使用 `lv_checkbox_set_static_text(cb, txt)`。这样，只会存储一个指向 `txt` 的指针。当复选框存在时，不应取消分配文本。

Check, uncheck, disable (选中, 取消选中, 禁用)

You can manually check, un-check, and disable the Checkbox by using the common state add/clear function:

您可以使用通用状态添加/清除功能手动选中、取消选中和禁用复选框：

```
lv_obj_add_state(cb, LV_STATE_CHECKED); /*Make the checkbox checked*/
lv_obj_clear_state(cb, LV_STATE_CHECKED); /*MAKE the checkbox unchecked*/
lv_obj_add_state(cb, LV_STATE_CHECKED | LV_STATE_DISABLED); /*Make the checkbox
↪checked and disabled*/
```

Events (事件)

- LV_EVENT_VALUE_CHANGED Sent when the checkbox is toggled.
- LV_EVENT_DRAW_PART_BEGIN and LV_EVENT_DRAW_PART_END are sent for the following types:
 - LV_CHECKBOX_DRAW_PART_BOX The tickbox of the checkbox
 - * part: LV_PART_INDICATOR
 - * draw_area: the area of the tickbox
 - * rect_dsc

See the events of the *Base object* too.

Learn more about *Events*.

- LV_EVENT_VALUE_CHANGED 当复选框被切换时发送。
- 为以下类型发送 LV_EVENT_DRAW_PART_BEGIN 和 LV_EVENT_DRAW_PART_END:
 - LV_CHECKBOX_DRAW_PART_BOX 复选框的勾选框
 - * 部分: LV_PART_INDICATOR
 - * draw_area: 勾选框的区域 -rect_dsc

参见*Base object* 的事件。

详细了解*事件*。

Keys (按键)

The following *Keys* are processed by the 'Buttons':

- LV_KEY_RIGHT/UP Go to toggled state if toggling is enabled
- LV_KEY_LEFT/DOWN Go to non-toggled state if toggling is enabled
- LV_KEY_ENTER Clicks the checkbox and toggles it

Note that, as usual, the state of LV_KEY_ENTER is translated to LV_EVENT_PRESSED/PRESSING/RELEASED etc.

Learn more about *Keys*.

以下键由“按钮”处理：

- LV_KEY_RIGHT/UP 如果启用切换，则转到切换状态
- LV_KEY_LEFT/DOWN 如果启用切换，则转到非切换状态

- LV_KEY_ENTER 单击复选框并切换它

请注意，像往常一样，“LV_KEY_ENTER”的状态被转换为“LV_EVENT_PRESSED/PRESSING/RELEASED”等。

了解有关[Keys](#) 的更多信息。

Example

C

Simple Checkboxes

```
#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(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        const char * txt = lv_checkbox_get_text(obj);
        const char * state = lv_obj_get_state(obj) & LV_STATE_CHECKED ? "Checked" :
        ↪"Unchecked";
        LV_LOG_USER("%s: %s", txt, state);
    }
}

void lv_example_checkbox_1(void)
{
    lv_obj_set_flex_flow(lv_scr_act(), LV_FLEX_FLOW_COLUMN);
    lv_obj_set_flex_align(lv_scr_act(), LV_FLEX_ALIGN_CENTER, LV_FLEX_ALIGN_START, LV_
    ↪FLEX_ALIGN_CENTER);

    lv_obj_t * cb;
    cb = lv_checkbox_create(lv_scr_act());
    lv_checkbox_set_text(cb, "Apple");
    lv_obj_add_event_cb(cb, event_handler, LV_EVENT_ALL, NULL);

    cb = lv_checkbox_create(lv_scr_act());
    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_scr_act());
    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_scr_act());
    lv_obj_add_state(cb, LV_STATE_CHECKED | 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);
}

```

(下页继续)

(续上页)

```
#endif
```

```
def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
        print("State: %s" % ("Checked" if obj.is_checked() else "Unchecked"))

cb = lv.cb(lv.scr_act())
cb.set_text("I agree to terms and conditions.")
cb.align(None, lv.ALIGN.CENTER, 0, 0)
cb.set_event_cb(event_handler)
```

MicroPython

No examples yet.

API

Functions

`lv_obj_t *lv_checkbox_create(lv_obj_t *parent)`

Create a check box object

参数 `parent` -- pointer to an object, it will be the parent of the new button

返回 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.

参数

- `cb` -- 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.

参数

- `cb` -- pointer to a check box
- `txt` -- the text of the check box. NULL to refresh with the current text.

`const char *lv_checkbox_get_text(const lv_obj_t *obj)`

Get the text of a check box

参数 `cb` -- pointer to check box object

返回 pointer to the text of the check box

Variables

```
const lv_obj_class_t lv_checkbox_class
struct lv_checkbox_t
```

Public Members

```
lv_obj_t obj
char *txt
uint32_t static_txt
```

6.2.7 Drop-down list (下拉列表) (**lv_dropdown**)

Overview (概述)

The drop-down list allows the user to select one value from a list.

The drop-down list is closed by default and displays a single value or a predefined text. When activated (by click on the drop-down list), a list is created from which the user may select one option. When the user selects a new value, the list is deleted again.

The Drop-down list is added to the default group (if it is set). Besides the Drop-down list is an editable object to allow selecting an option with encoder navigation too.

下拉列表允许用户从列表中选择一个值。

下拉列表默认关闭并显示单个值或预定义文本。激活后（通过单击下拉列表），将创建一个列表，用户可以从中选择一个选项。当用户选择一个新值时，该列表将再次被删除。

下拉列表将添加到默认组（如果已设置）。除了下拉列表是一个可编辑的对象，也允许选择带有编码器导航的选项。

Parts and Styles (部件和样式)

The Dropdown widget is built from the elements: "button" and "list" (both not related to the button and list widgets)

下拉小部件由以下元素构建：“按钮” 和 “列表”（均与按钮和列表小部件无关）

Button (按钮)

- **LV_PART_MAIN** The background of the button. Uses the typical background properties and text properties for the text on it.
- **LV_PART_INDICATOR** Typically an arrow symbol that can be an image or a text (**LV_SYMBOL**).

The button goes to **LV_STATE_CHECKED** when its opened.

- **LV_PART_MAIN** 按钮的背景。对其上的文本使用典型的背景属性和文本属性。
- **LV_PART_INDICATOR** 通常是一个箭头符号，可以是图像或文本 (**LV_SYMBOL**)。

按钮在打开时会转到 “**LV_STATE_CHECKED**”。

List (列表)

- **LV_PART_MAIN** The list itself. Uses the typical background properties. `max_height` can be used to limit the height of the list.
- **LV_PART_SCROLLBAR** The scrollbar background, border, shadow properties and width (for its own width) and right padding for the spacing on the right.
- **LV_PART_SELECTED** Refers to the currently pressed, checked or pressed+checked option. Also uses the typical background properties.

As list does not exist when the drop-down list is closed it's not possible to simply add styles to it. Instead the following should be done:

1. Add an event handler to the button for `LV_EVENT_VALUE_CHANGED` (triggered when the list is opened/closed)
2. Use `lv_obj_t * list = lv_dropdown_get_list(dropdown)`
3. `if(list != NULL) /*Add the styles to the list*/`

Alternatively the theme can be extended with the new styles.

- **LV_PART_MAIN** 列表本身。使用典型的背景属性。`max_height` 可用于限制列表的高度。
- **LV_PART_SCROLLBAR** 滚动条背景、边框、阴影属性和宽度（对于它自己的宽度）以及右侧间距的右侧填充。
- **LV_PART_SELECTED** 指的是当前按下、选中或按下 + 选中的选项。还使用典型的背景属性。

由于下拉列表关闭时列表不存在，因此无法简单地向其添加样式。相反，应执行以下操作：

1. 为 `LV_EVENT_VALUE_CHANGED` 的按钮添加一个事件处理程序（在列表打开/关闭时触发）
2. 使用 `lv_obj_t * list = lv_dropdown_get_list(dropdown)`
2. `if(list != NULL) /* 将样式添加到列表中 */`

或者，可以使用新样式扩展主题。

Usage (用法)

Overview (概述)

Set options (设置选项)

Options are passed to the drop-down list as a string with `lv_dropdown_set_options(dropdown, options)`. Options should be separated by `\n`. For example: "First\nSecond\nThird". This string will be saved in the drop-down list, so it can be used in a local variable.

The `lv_dropdown_add_option(dropdown, "New option", pos)` function inserts a new option to `pos` index.

To save memory the options can be set from a static(constant) string too with `lv_dropdown_set_static_options(dropdown, options)`. In this case the options string should be alive while the drop-down list exists and `lv_dropdown_add_option` can't be used.

You can select an option manually with `lv_dropdown_set_selected(dropdown, id)`, where `id` is the index of an option.

选项作为带有 `lv_dropdown_set_options(dropdown, options)` 的字符串传递给下拉列表。选项应该用 `\n` 分隔。例如: "First\nSecond\nThird"。该字符串将保存在下拉列表中，因此它可以保存在局部变量中。

`lv_dropdown_add_option(dropdown, "New option", pos)` 函数向 `pos` 索引插入一个新选项。

为了节省内存，选项也可以使用 `lv_dropdown_set_static_options(dropdown, options)` 从静态（常量）字符串中设置。在这种情况下，当下拉列表存在且不能使用 `lv_dropdown_add_option` 时，选项字符串应该是活动的。

您可以使用 `lv_dropdown_set_selected(dropdown, id)` 手动选择一个选项，其中 `id` 是一个选项的索引。

Get selected option (获取选择的选项)

要获取所选选项的 `index`，使用 `lv_dropdown_get_selected(dropdown)`。

`lv_dropdown_get_selected_str(dropdown, buf, buf_size)` 将所选选项的 `name` 复制到 `buf`。

要获取所选选项的 `index`，使用 `lv_dropdown_get_selected(dropdown)`。

`lv_dropdown_get_selected_str(dropdown, buf, buf_size)` 将所选选项的 `name` 复制到 `buf`。

Direction (方向)

列表可以在任何一侧创建。默认的 `LV_DIR_BOTTOM` 可以通过 `lv_dropdown_set_dir(dropdown, LV_DIR_LEFT/RIGHT/UP/BOTTOM)` 函数进行修改。

如果列表垂直于屏幕之外，它将与边缘对齐。

列表可以在任何一侧创建。默认的 `LV_DIR_BOTTOM` 可以通过 `lv_dropdown_set_dir(dropdown, LV_DIR_LEFT/RIGHT/UP/BOTTOM)` 函数进行修改。

如果列表垂直于屏幕之外，它将与边缘对齐。

Symbol (符号)

一个符号（通常是一个箭头）可以添加到下拉列表中，使用 `lv_dropdown_set_symbol(dropdown, LV_SYMBOL_...)`。

如果下拉列表的方向是 `LV_DIR_LEFT`，符号将显示在左侧，否则显示在右侧。

可以使用 `lv_dropdown_set_symbol(dropdown, LV_SYMBOL_...)` 将符号（通常是箭头）添加到下拉列表中。

如果下拉列表的方向是 `LV_DIR_LEFT`，符号将显示在左侧，否则显示在右侧。

Show selected (显示选中)

主要部分可以显示所选选项或静态文本。如果使用 `lv_dropdown_set_text(dropdown, "Some text")` 设置静态，无论选择哪个选项，它都会显示。如果文本为“NULL”，则所选选项将显示在按钮上。

主要部分可以显示所选选项或静态文本。如果使用 `lv_dropdown_set_text(dropdown, "Some text")` 设置静态，无论选择哪个选项，它都会显示。如果文本为“NULL”，则所选选项将显示在按钮上。

Manually open/close (手动打开/关闭)

To manually open or close the drop-down list the `lv_dropdown_open/close(dropdown)` function can be used.
要手动打开或关闭下拉列表，可以使用 `lv_dropdown_open/close(dropdown)` 函数。

Events (事件)

Apart from the [Generic events](#), the following [Special events](#) are sent by the drop-down list:

- `LV_EVENT_VALUE_CHANGED` Sent when the new option is selected or the list is opened/closed.

See the events of the [Base object](#) too.

Learn more about [Events](#).

除了通用事件，下拉列表发送以下特殊事件：

- `LV_EVENT_VALUE_CHANGED` 在选择新选项或打开/关闭列表时发送。

参见[Base object](#) 的事件。

详细了解[事件](#)。

Keys (按键)

- `LV_KEY_RIGHT/DOWN` Select the next option.
- `LV_KEY_LEFT/UP` Select the previous option.
- `LY_KEY_ENTER` Apply the selected option (Sends `LV_EVENT_VALUE_CHANGED` event and closes the drop-down list).

Learn more about [Keys](#).

- `LV_KEY_RIGHT/DOWN` 选择下一个选项。
- `LV_KEY_LEFT/UP` 选择上一个选项。
- `LY_KEY_ENTER` 应用选择的选项（发送 `LV_EVENT_VALUE_CHANGED` 事件并关闭下拉列表）。

了解有关[Keys](#) 的更多信息。

Example

C

Simple Drop down list

```
#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(e);
    if(code == LV_EVENT_VALUE_CHANGED) {
        char buf[32];
    }
}
```

(下页继续)

(续上页)

```

    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_scr_act());
    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

```

```

def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
        option = " "*10 # should be large enough to store the option
        obj.get_selected_str(option, len(option))
        # .strip() removes trailing spaces
        print("Option: \"%s\" % option.strip())

    # Create a drop down list
    dlist = lv.ddlist(lv.scr_act())
    dlist.set_options("\n".join([
        "Apple",
        "Banana",
        "Orange",
        "Melon",
        "Grape",
        "Raspberry"]))

    dlist.set_fix_width(150)
    dlist.set_draw_arrow(True)
    dlist.align(None, lv.ALIGN.IN_TOP_MID, 0, 20)
    dlist.set_event_cb(event_handler)

```

Drop down in four directions

```
#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\n"
                               "Grape\n"
                               "Raspberry";

    lv_obj_t * dd;
    dd = lv_dropdown_create(lv_scr_act());
    lv_dropdown_set_options_static(dd, opts);
    lv_obj_align(dd, LV_ALIGN_TOP_MID, 0, 10);

    dd = lv_dropdown_create(lv_scr_act());
    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_scr_act());
    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_scr_act());
    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

```

```
# Create a drop UP list by applying auto realign

# Create a drop down list
ddlist = lv.ddlist(lv.scr_act())
ddlist.set_options("\n".join([
    "Apple",
    "Banana",
    "Orange",
    "Melon",
    "Grape",
    "Raspberry"]))

```

(下页继续)

(续上页)

```

ddlist.set_fix_width(150)
ddlist.set_fix_height(150)
ddlist.set_draw_arrow(True)

# Enable auto-realign when the size changes.
# It will keep the bottom of the ddlist fixed
ddlist.set_auto_realign(True)

# It will be called automatically when the size changes
ddlist.align(None, lv.ALIGN.IN_BOTTOM_MID, 0, -20)

```

Menu

```

#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(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_scr_act());
    lv_obj_align(dropdown, LV_ALIGN_TOP_LEFT, 10, 10);
    lv_dropdown_set_options(dropdown, "New project\n"
                            "New file\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_IMG_DECLARE(img_caret_down)
    lv_dropdown_set_symbol(dropdown, &img_caret_down);
    lv_obj_set_style_transform_angle(dropdown, 180, 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);
}

```

(下页继续)

(续上页)

```
#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/widgets/dropdown/lv_example_dropdown_3.py
```

MicroPython

No examples yet.

API

Functions

LV_EXPORT_CONST_INT(LV_DROPDOWN_POS_LAST)

lv_obj_t ***lv_dropdown_create**(*lv_obj_t* *parent)

Create a drop-down list objects

参数 **parent** -- pointer to an object, it will be the parent of the new drop-down list

返回 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 the selected option.

参数

- **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

参数

- **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.

参数

- **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.

参数

- **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.

参数 obj -- pointer to drop-down list object

void lv_dropdown_set_selected(*lv_obj_t* *obj, uint16_t sel_opt)
Set the selected option

参数

- **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

参数

- **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.

注解: 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

参数

- **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

参数

- **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

参数 obj -- pointer to a drop-down list object

返回 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.

参数 **obj** -- pointer to a drop-down list object

返回 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

参数 **obj** -- pointer to drop-down list object

返回

the options separated by '

'-s (E.g. "Option1\nOption2\nOption3")

`uint16_t lv_dropdown_get_selected(const lv_obj_t *obj)`

Get the index of the selected option

参数 **obj** -- pointer to drop-down list object

返回 index of the selected option (0 ... number of option - 1);

`uint16_t lv_dropdown_get_option_cnt(const lv_obj_t *obj)`

Get the total number of options

参数 **obj** -- pointer to drop-down list object

返回 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

参数

- **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.

`const char *lv_dropdown_get_symbol(lv_obj_t *obj)`

Get the symbol on the drop-down list. Typically a down caret or arrow.

参数 **obj** -- pointer to drop-down list object

返回 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

参数 **obj** -- pointer to drop-down list object

返回 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

参数 **obj** -- pointer to a drop-down list object

返回 LV_DIR_LEFT/RIGHT/TOP/BOTTOM

`void lv_dropdown_open(lv_obj_t *dropdown_obj)`

Open the drop-down list

参数 **obj** -- pointer to drop-down list object

void **lv_dropdown_close**(*lv_obj_t* *obj)
 Close (Collapse) the drop-down list
 参数 **obj** -- pointer to drop-down list object

Variables

```
const lv_obj_class_t lv_dropdown_class  

const lv_obj_class_t lv_dropdownlist_class  

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 a '
 ' separated list

uint16_t **option_cnt**

Number of options

uint16_t **sel_opt_id**

Index of the currently selected option

uint16_t **sel_opt_id_orig**

Store the original index on focus

uint16_t **pr_opt_id**

Index of the currently pressed option

lv_dir_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**

6.2.8 Image (图象) (lv_img)

Overview (概述)

Images are the basic object to display images from flash (as arrays) or from files. Images can display symbols (LV_SYMBOL_...) too.

Using the [Image decoder interface](#) custom image formats can be supported as well.

图像是显示来自闪存（作为数组）或来自文件的图像的基本对象。图像也可以显示符号 (LV_SYMBOL_...).

使用[图像解码器接口](#) 也可以支持自定义图像格式。

Parts and Styles (部件和风格)

- LV_PART_MAIN A background rectangle that uses the typical background style properties and the image itself using the image style properties.
- LV_PART_MAIN 使用典型背景样式属性的背景矩形和使用图像样式属性的图像本身。

Usage (用法)

Image source (图片来源)

To provide maximum flexibility, the source of the image can be:

- a variable in code (a C array with the pixels).
- a file stored externally (e.g. on an SD card).
- a text with *Symbols*.

To set the source of an image, use `lv_img_set_src(img, src)`.

To generate a pixel array from a PNG, JPG or BMP image, use the [Online image converter tool](#) and set the converted image with its pointer: `lv_img_set_src(img1, &converted_img_var)`; To make the variable visible in the C file, you need to declare it with `LV_IMG_DECLARE(converted_img_var)`.

为了提供最大的灵活性，图像的来源可以是：

- 代码中的变量（带有像素的 C 数组）。
- 外部存储的文件（例如在 SD 卡上）。
- 带有 *Symbols* 的文本。

要设置图像的来源，请使用 `lv_img_set_src(img, src)`。

要从 PNG、JPG 或 BMP 图像生成像素数组，请使用 [在线图像转换工具](#) 并使用其指针设置转换后的图像：`lv_img_set_src(img1, &converted_img_var)`；要使该变量在 C 文件中可见，您需要使用 `LV_IMG_DECLARE(converted_img_var)` 声明它。

To use external files, you also need to convert the image files using the online converter tool but now you should select the binary output format. You also need to use LVGL's file system module and register a driver with some functions for the basic file operation. Go to the [File system](#) to learn more. To set an image sourced from a file, use `lv_img_set_src(img, "S:folder1/my_img.bin")`.

You can also set a symbol similarly to [Labels](#). In this case, the image will be rendered as text according to the *font* specified in the style. It enables to use of light-weight monochrome "letters" instead of real images. You can set symbol like `lv_img_set_src(img1, LV_SYMBOL_OK)`.

要使用外部文件，您还需要使用在线转换器工具转换图像文件，但现在您应该选择二进制输出格式。您还需要使用 LVGL 的文件系统模块，并为基本文件操作注册一个具有某些功能的驱动程序。转到[文件系统](#)了解更多信息。要设置来自文件的图像，请使用 `lv_img_set_src(img, "S:folder1/my_img.bin")`。

您还可以设置类似于[标签](#)的符号。在这种情况下，图像将根据样式中指定的 *font* 呈现为文本。它允许使用轻量级单色“字母”而不是真实图像。你可以设置像 `lv_img_set_src(img1, LV_SYMBOL_OK)` 这样的符号。

Label as an image (标签作为图象)

Images and labels are sometimes used to convey the same thing. For example, to describe what a button does. Therefore, images and labels are somewhat interchangeable, that is the images can display texts by using `LV_SYMBOL_DUMMY` as the prefix of the text. For example, `lv_img_set_src(img, LV_SYMBOL_DUMMY "Some text")`.

图像和标签有时用于传达相同的内容。例如，描述按钮的作用。因此，图像和标签在某种程度上是可以互换的，即图像可以通过使用“`LV_SYMBOL_DUMMY`”作为文本的前缀来显示文本。例如，`lv_img_set_src(img, LV_SYMBOL_DUMMY "Some text")`。

Transparency (透明度)

The internal (variable) and external images support 2 transparency handling methods:

- **Chroma-keying** - Pixels with `LV_COLOR_CHROMA_KEY` (*lv_conf.h*) color will be transparent.
- **Alpha byte** - An alpha byte is added to every pixel that contains the pixel's opacity

内部（变量）和外部图像支持 2 种透明度处理方法：

- **色度键控** - 具有 `LV_COLOR_CHROMA_KEY` (*lv_conf.h*) 颜色的像素将是透明的。
- **Alpha 字节** - 向每个包含像素不透明度的像素添加一个 Alpha 字节

Palette and Alpha index (调色板和 Alpha 索引)

Besides the *True color* (RGB) color format, the following formats are supported:

- **Indexed** - Image has a palette.
- **Alpha indexed** - Only alpha values are stored.

These options can be selected in the image converter. To learn more about the color formats, read the [Images](#) section.

除了 *True color* (RGB) 颜色格式外，还支持以下格式：

- 索引 - 图像有调色板。
- **Alpha 索引** - 仅存储 Alpha 值。

可以在图像转换器中选择这些选项。要了解有关颜色格式的更多信息，请阅读[图像](#)部分。

Recolor (重新着色)

A color can be mixed with every pixel of an image with a given intensity. This can be useful to show different states (checked, inactive, pressed, etc.) of an image without storing more versions of the same image. This feature can be enabled in the style by setting `img_recolor_opa` between `LV_OPA_TRANS` (no recolor, value: 0) and `LV_OPA_COVER` (full recolor, value: 255). The default value is `LV_OPA_TRANS` so this feature is disabled.

The color to mix is set by `img_recolor`.

颜色可以与具有给定强度的图像的每个像素混合。这对于显示图像的不同状态（选中、非活动、按下等）非常有用，而无需存储同一图像的更多版本。可以通过在“`LV_OPA_TRANS`”（无重新着色，值：0）和“`LV_OPA_COVER`”（完全重新着色，值：255）之间设置“`img_recolor_opa`”在样式中启用此功能。默认值为“`LV_OPA_TRANS`”，因此禁用此功能。

要混合的颜色由 `img_recolor` 设置。

Auto-size (自动大小)

If the width or height of the image object is set to `LV_SIZE_CONTENT` the object's size will be set according to the size of the image source in the respective direction.

如果图像对象的宽度或高度设置为 `LV_SIZE_CONTENT`，则对象的大小将根据图像源在相应方向上的大小设置。

Mosaic (马赛克)

If the object's size is greater than the image size in any directions, then the image will be repeated like a mosaic. This allows creation a large image from only a very narrow source. For example, you can have a 300×5 image with a special gradient and set it as a wallpaper using the mosaic feature.

如果对象的大小在任何方向上都大于图像大小，则图像将像马赛克一样重复。这允许仅从非常窄的源创建大图像。例如，您可以使用带有特殊渐变的 300×5 图像，并使用马赛克功能将其设置为墙纸。

Offset (偏移)

With `lv_img_set_offset_x(img, x_ofs)` and `lv_img_set_offset_y(img, y_ofs)`, you can add some offset to the displayed image. Useful if the object size is smaller than the image source size. Using the offset parameter a [Texture atlas](#) or a "running image" effect can be created by [Animating](#) the x or y offset.

使用 `lv_img_set_offset_x(img, x_ofs)` 和 `lv_img_set_offset_y(img, y_ofs)`，您可以为显示的图像添加一些偏移量。如果对象大小小于图像源大小，则很有用。使用偏移参数 [Texture atlas](#) 或“运行图像”效果可以通过[Animating](#) x 或 y 偏移创建。

Transformations (转换)

Using the `lv_img_set_zoom(img, factor)` the images will be zoomed. Set `factor` to 256 or `LV_IMG_ZOOM_NONE` to disable zooming. A larger value enlarges the images (e.g. 512 double size), a smaller value shrinks it (e.g. 128 half size). Fractional scale works as well. E.g. 281 for 10% enlargement.

To rotate the image use `lv_img_set_angle(img, angle)`. Angle has 0.1 degree precision, so for 45.8° set 458.

The `transform_zoom` and `transform_angle` style properties are also used to determine the final zoom and angle.

By default, the pivot point of the rotation is the center of the image. It can be changed with `lv_img_set_pivot(img, pivot_x, pivot_y)`. `0;0` is the top left corner.

使用 `lv_img_set_zoom(img, factor)` 图像将被缩放。将 `factor` 设置为 256 或 `LV_IMG_ZOOM_NONE` 以禁用缩放。较大的值会放大图像（例如“512”双倍尺寸），较小的值会缩小图像（例如“128”半尺寸）。分数尺度也有效。例如。`281` 放大 10%。

要旋转图像，请使用 `lv_img_set_angle(img, angle)`。角度的精度为 0.1 度，因此对于 45.8° 设置 458。

`transform_zoom` 和 `transform_angle` 样式属性也用于确定最终的缩放和角度。

默认情况下，旋转的轴心点是图像的中心。它可以通过 `lv_img_set_pivot(img, pivot_x, pivot_y)` 改变。`0;0` 是左上角。

The quality of the transformation can be adjusted with `lv_img_set_antialias(img, true/false)`. With enabled anti-aliasing the transformations are higher quality but slower.

The transformations require the whole image to be available. Therefore indexed images (`LV_IMG_CF_INDEXED_...`), alpha only images (`LV_IMG_CF_ALPHA_...`) or images from files can not be transformed. In other words transformations work only on true color images stored as C array, or if a custom [Image decoder](#) returns the whole image.

Note that the real coordinates of image objects won't change during transformation. That is `lv_obj_get_width/height/x/y()` will return the original, non-zoomed coordinates.

转换的质量可以通过 `lv_img_set_antialias(img, true/false)` 来调整。启用抗锯齿后，转换质量更高但速度更慢。

转换需要整个图像可用。因此，索引图像 (`LV_IMG_CF_INDEXED_...`)、仅 alpha 图像 (`LV_IMG_CF_ALPHA_...`) 或来自文件的图像无法转换。换句话说，转换仅适用于存储为 C 数组的真彩色图像，或者如果自定义 [图像解码器](#) 返回整个图像。

请注意，图像对象的真实坐标在转换过程中不会改变。即 `lv_obj_get_width/height/x/y()` 将返回原始的、未缩放的坐标。

Size mode (尺寸模式)

By default if the image is zoom or rotated the real coordinates of the image object are not changed. The larger content simply overflows the object's boundaries. It also means the layouts are not affected by the transformations.

If you need the object size to be updated to the transformed size set `lv_img_set_size_mode(img, LV_IMG_SIZE_MODE_REAL)`. (The previous mode is the default and called `LV_IMG_SIZE_MODE_VIRTUAL`). In this case if the width/height of the object is set to `LV_SIZE_CONTENT` the object's size will be set to the zoomed and rotated size. If an explicit size is set then the overflowing content will be cropped.

默认情况下，如果图像被缩放或旋转，图像对象的真实坐标不会改变。较大的内容只是溢出对象的边界。这也意味着布局不受转换的影响。

如 果 您 需 要 将 对 象 大 小 更 新 为 转 换 后 的 大 小 集 `lv_img_set_size_mode(img, LV_IMG_SIZE_MODE_REAL)`。 (之前的模式是默认模式，称为 `LV_IMG_SIZE_MODE_VIRTUAL`) 。 在这种情况下，如果对象的宽度/高度设置为“`LV_SIZE_CONTENT`”，则对象的大小将设置为缩放和旋转后的大小。如果设置了明确的大小，那么溢出的内容将被裁剪。

Events (事件)

No special events are sent by image objects.

See the events of the [Base object](#) too.

Learn more about [Events](#).

图像对象不发送特殊事件。

参见[Base object](#) 的事件。

详细了解[事件](#)。

Keys (按键)

No *Keys* are processed by the object type.

Learn more about [Keys](#).

对象类型不处理 *Keys*。

了解有关[Keys](#) 的更多信息。

Example

C

Image from variable and symbol

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

void lv_example_img_1(void)
{
    LV_IMG_DECLARE(img_cogwheel_argb);
    lv_obj_t * img1 = lv_img_create(lv_scr_act());
```

(下页继续)

(续上页)

```

lv_img_set_src(img1, &img_cogwheel_argb);
lv_obj_align(img1, LV_ALIGN_CENTER, 0, -20);
lv_obj_set_size(img1, 200, 200);

lv_obj_t * img2 = lv_img_create(lv_scr_act());
lv_img_set_src(img2, LV_SYMBOL_OK "Accept");
lv_obj_align_to(img2, img1, LV_ALIGN_OUT_BOTTOM_MID, 0, 20);
}

#endif

```

```

from imagetools import get_png_info, open_png

# Register PNG image decoder
decoder = lv.img.decoder_create()
decoder.info_cb = get_png_info
decoder.open_cb = open_png

# Create a screen with a draggable image

with open('cogwheel.png', 'rb') as f:
    png_data = f.read()

png_img_dsc = lv.img_dsc_t({
    'data_size': len(png_data),
    'data': png_data
})

scr = lv.scr_act()

# Create an image on the left using the decoder

# lv.img.cache_set_size(2)
img1 = lv.img(scr)
img1.align(scr, lv.ALIGN.CENTER, 0, -20)
img1.set_src(png_img_dsc)

img2 = lv.img(scr)
img2.set_src(lv.SYMBOL.OK + "Accept")
img2.align(img1, lv.ALIGN.OUT_BOTTOM_MID, 0, 20)

```

Image recoloring

```

#include "../../lv_examples.h"
#if LV_USE_IMG && 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;

/**

```

(下页继续)

(续上页)

```

* Demonstrate runtime image re-coloring
*/
void lv_example_img_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_IMG_DECLARE(img_cogwheel_argb)
    img1 = lv_img_create(lv_scr_act());
    lv_img_set_src(img1, &img_cogwheel_argb);
    lv_obj_align(img1, LV_ALIGN_RIGHT_MID, -20, 0);

    lv_event_send(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(lv_slider_get_value(red_slider), lv_slider_get_
    ↪value(green_slider), lv_slider_get_value(blue_slider));
    lv_opa_t intense = lv_slider_get_value(intense_slider);
    lv_obj_set_style_img_recolor_opa(img1, intense, 0);
    lv_obj_set_style_img_recolor(img1, color, 0);
}

static lv_obj_t * create_slider(lv_color_t color)
{
    lv_obj_t * slider = lv_slider_create(lv_scr_act());
    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

```

Error encountered **while** trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_ ↪lvgl_docs_8.x/examples/widgets/img/lv_example_img_2.py

Rotate and zoom

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

static void set_angle(void * img, int32_t v)
{
    lv_img_set_angle(img, v);
}

static void set_zoom(void * img, int32_t v)
{
    lv_img_set_zoom(img, v);
}

/**
 * Show transformations (zoom and rotation) using a pivot point.
 */
void lv_example_img_3(void)
{
    LV_IMG_DECLARE(img_cogwheel_argb);

    /*Now create the actual image*/
    lv_obj_t * img = lv_img_create(lv_scr_act());
    lv_img_set_src(img, &img_cogwheel_argb);
    lv_obj_align(img, LV_ALIGN_CENTER, 50, 50);
    lv_img_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_time(&a, 5000);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);

    lv_anim_set_exec_cb(&a, set_zoom);
    lv_anim_set_values(&a, 128, 256);
    lv_anim_set_playback_time(&a, 3000);
    lv_anim_start(&a);
}

#endif
```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/img/lv_example_img_3.py

Image offset and styling

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

static void ofs_y_anim(void * img, int32_t v)
{
    lv_img_set_offset_y(img, v);
}

/**
 * Image styling and offset
 */
void lv_example_img_4(void)
{
    LV_IMG_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_img_recolor_opa(&style, LV_OPA_COVER);
    lv_style_set_img_recolor(&style, lv_color_black());

    lv_obj_t * img = lv_img_create(lv_scr_act());
    lv_obj_add_style(img, &style, 0);
    lv_img_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);
    lv_anim_set_time(&a, 3000);
    lv_anim_set_playback_time(&a, 500);
    lv_anim_set_repeat_count(&a, LV_ANIM_REPEAT_INFINITE);
    lv_anim_start(&a);
}

#endif
```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/img/lv_example_img_4.py

MicroPython

No examples yet.

API

Functions

`lv_obj_t *lv_img_create(lv_obj_t *parent)`

Create a image objects

参数 **parent** -- pointer to an object, it will be the parent of the new image

返回 pointer to the created image

`void lv_img_set_src(lv_obj_t *obj, const void *src)`

Set the image data to display on the the object

参数

- **obj** -- pointer to an image object
- **src_img** -- 1) pointer to an `lv_img_dsc_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_img_set_offset_x(lv_obj_t *obj, lv_coord_t x)`

Set an offset for the source of an image so the image will be displayed from the new origin.

参数

- **obj** -- pointer to an image
- **x** -- the new offset along x axis.

`void lv_img_set_offset_y(lv_obj_t *obj, lv_coord_t y)`

Set an offset for the source of an image. so the image will be displayed from the new origin.

参数

- **obj** -- pointer to an image
- **y** -- the new offset along y axis.

`void lv_img_set_angle(lv_obj_t *obj, int16_t angle)`

Set the rotation angle of the image. The image will be rotated around the set pivot set by `lv_img_set_pivot()`

参数

- **obj** -- pointer to an image object
- **angle** -- rotation angle in degree with 0.1 degree resolution (0..3600: clock wise)

`void lv_img_set_pivot(lv_obj_t *obj, lv_coord_t x, lv_coord_t y)`

Set the rotation center of the image. The image will be rotated around this point

参数

- **obj** -- pointer to an image object
- **x** -- rotation center x of the image
- **y** -- rotation center y of the image

void **lv_img_set_zoom**(*lv_obj_t* *obj, uint16_t zoom)

void **lv_img_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.

参数

- **obj** -- pointer to an image object
- **antialias** -- true: anti-aliased; false: not anti-aliased

const void ***lv_img_get_src**(*lv_obj_t* *obj)

Get the source of the image

参数 **obj** -- pointer to an image object

返回 the image source (symbol, file name or ::lv-img_dsc_t for C arrays)

lv_coord_t **lv_img_get_offset_x**(*lv_obj_t* *obj)

Get the offset's x attribute of the image object.

参数 **img** -- pointer to an image

返回 offset X value.

lv_coord_t **lv_img_get_offset_y**(*lv_obj_t* *obj)

Get the offset's y attribute of the image object.

参数 **obj** -- pointer to an image

返回 offset Y value.

uint16_t **lv_img_get_angle**(*lv_obj_t* *obj)

Get the rotation angle of the image.

参数 **obj** -- pointer to an image object

返回 rotation angle in 0.1 degrees (0..3600)

void **lv_img_get_pivot**(*lv_obj_t* *obj, *lv_point_t* *pivot)

Get the pivot (rotation center) of the image.

参数

- **img** -- pointer to an image object
- **pivot** -- store the rotation center here

uint16_t **lv_img_get_zoom**(*lv_obj_t* *obj)

Get the zoom factor of the image.

参数 **obj** -- pointer to an image object

返回 zoom factor (256: no zoom)

bool **lv_img_get_antialias**(*lv_obj_t* *obj)

Get whether the transformations (rotate, zoom) are anti-aliased or not

参数 **obj** -- pointer to an image object

返回 true: anti-aliased; false: not anti-aliased

Variables

```
const lv_obj_class_t lv_img_class
struct lv_img_t
```

Public Members

```
lv_obj_t obj
const void *src
lv_point_t offset
lv_coord_t w
lv_coord_t h
uint16_t angle
lv_point_t pivot
uint16_t zoom
uint8_t src_type
uint8_t cf
uint8_t antialias
```

6.2.9 Label (标签) (**lv_label**)

Overview (概述)

A label is the basic object type that is used to display text.

标签是用于显示文本的基本对象类型。

Parts and Styles (部件和风格)

- **LV_PART_MAIN** Uses all the typical background properties and the text properties. The padding values can be used to add space between the text and the background.
- **LV_PART_SCROLLBAR** The scrollbar that is shown when the text is larger than the widget's size.
- **LV_PART_SELECTED** Tells the style of the *selected text*. Only **text_color** and **bg_color** style properties can be used.
- **LV_PART_MAIN** 使用所有典型的背景属性和文本属性。填充值可用于在文本和背景之间添加空间。
- **LV_PART_SCROLLBAR** 当文本大于小部件的大小时显示的滚动条。
- **LV_PART_SELECTED** 告诉所选文本的样式。只能使用 **text_color** 和 **bg_color** 样式属性。

Usage (用法)

Set text (设置文本)

You can set the text on a label at runtime with `lv_label_set_text(label, "New text")`. This will allocate a buffer dynamically, and the provided string will be copied into that buffer. Therefore, you don't need to keep the text you pass to `lv_label_set_text` in scope after that function returns.

With `lv_label_set_text_fmt(label, "Value: %d", 15)` printf formatting can be used to set the text.

您可以在运行时使用 `lv_label_set_text(label, "New text")` 设置标签上的文本。这将动态分配一个缓冲区，并且提供的字符串将被复制到该缓冲区中。因此，在该函数返回后，您不需要将传递给 `lv_label_set_text` 的文本保留在作用域中。

使用 `lv_label_set_text_fmt(label, "Value: %d", 15)` printf 格式可用于设置文本。

Labels are able to show text from a static character buffer. To do so, use `lv_label_set_text_static(label, "Text")`. In this case, the text is not stored in the dynamic memory and the given buffer is used directly instead. This means that the array can't be a local variable which goes out of scope when the function exits. Constant strings are safe to use with `lv_label_set_text_static` (except when used with `LV_LABEL_LONG_DOT`, as it modifies the buffer in-place), as they are stored in ROM memory, which is always accessible.

标签能够显示来自静态字符缓冲区的文本。为此，请使用 `lv_label_set_text_static(label, "Text")`。在这种情况下，文本不存储在动态内存中，而是直接使用给定的缓冲区。这意味着数组不能是在函数退出时超出范围的局部变量。常量字符串可以安全地与 `lv_label_set_text_static` 一起使用（除非与 `LV_LABEL_LONG_DOT` 一起使用，因为它会就地修改缓冲区），因为它们存储在 ROM 内存中，始终可以访问。

Newline (新行)

Newline characters are handled automatically by the label object. You can use \n to make a line break. For example: "line1\nline2\n\nline4"

换行符由标签对象自动处理。您可以使用 \n 来换行。例如: "line1\nline2\n\nline4"

Long modes (长模式)

By default, the width and height of the label is set to `LV_SIZE_CONTENT`. Therefore the size of the label is automatically expanded to the text size. Otherwise, if the width or height are explicitly set (using e.g. `lv_obj_set_width` or a layout), the lines wider than the label's width can be manipulated according to several long mode policies. Similarly, the policies can be applied if the height of the text is greater than the height of the label.

- `LV_LABEL_LONG_WRAP` Wrap too long lines. If the height is `LV_SIZE_CONTENT` the label's height will be expanded, otherwise the text will be clipped. (Default)
- `LV_LABEL_LONG_DOT` Replaces the last 3 characters from bottom right corner of the label with dots (.)
- `LV_LABEL_LONG_SCROLL` If the text is wider than the label scroll it horizontally back and forth. If it's higher, scroll vertically. Only one direction is scrolled and horizontal scrolling has higher precedence.
- `LV_LABEL_LONG_SCROLL_CIRCULAR` If the text is wider than the label scroll it horizontally continuously. If it's higher, scroll vertically. Only one direction is scrolled and horizontal scrolling has higher precedence.
- `LV_LABEL_LONG_CLIP` Simply clip the parts of the text outside of the label.

默认情况下，标签的宽度和高度设置为 `LV_SIZE_CONTENT`。因此标签的大小会自动扩展到文本大小。否则，如果显式设置宽度或高度（使用例如 `lv_obj_set_width` 或布局），则可以根据几种长模式策略来操作比标签宽度更宽的行。类似地，如果文本的高度大于标签的高度，则可以应用策略。

- `LV_LABEL_LONG_WRAP` 换行太长。如果高度为 `LV_SIZE_CONTENT`, 标签的高度将被扩展, 否则文本将被剪裁。(默认)
- `LV_LABEL_LONG_DOT` 将标签右下角的最后 3 个字符替换为点(,.)
- `LV_LABEL_LONG_SCROLL` 如果文本比标签宽, 则水平来回滚动它。如果它更高, 请垂直滚动。只滚动一个方向, 水平滚动的优先级更高。
- `LV_LABEL_LONG_SCROLL_CIRCULAR` 如果文本比标签宽, 则水平滚动它。如果它更高, 请垂直滚动。只滚动一个方向, 水平滚动的优先级更高。
- `LV_LABEL_LONG_CLIP` 只需剪掉标签外的文本部分。

You can specify the long mode with `lv_label_set_long_mode(label, LV_LABEL_LONG_...)`

Note that `LV_LABEL_LONG_DOT` manipulates the text buffer in-place in order to add/remove the dots. When `lv_label_set_text` or `lv_label_set_array_text` are used, a separate buffer is allocated and this implementation detail is unnoticed. This is not the case with `lv_label_set_text_static`. The buffer you pass to `lv_label_set_text_static` must be writable if you plan to use `LV_LABEL_LONG_DOT`.

您可以使用 `lv_label_set_long_mode(label, LV_LABEL_LONG_...)` 指定长模式

请注意, `LV_LABEL_LONG_DOT` 就地操作文本缓冲区以添加/删除点。当使用 `lv_label_set_text` 或 `lv_label_set_array_text` 时, 会分配一个单独的缓冲区, 并且不会注意到此实现细节。`lv_label_set_text_static` 不是这种情况。如果你打算使用 `LV_LABEL_LONG_DOT`, 你传递给 `lv_label_set_text_static` 的缓冲区必须是可写的。

Text recolor (文本重新着色)

In the text, you can use commands to recolor parts of the text. For example: "Write a #ff0000 red# word". This feature can be enabled individually for each label by `lv_label_set_recolor()` function.

在文本中, 您可以使用命令对文本的某些部分重新着色。例如: “写一个 #ff0000 red# 字”。可以通过 `lv_label_set_recolor()` 函数为每个标签单独启用此功能。

Text selection (文本选择)

If enabled by `LV_LABEL_TEXT_SELECTION` part of the text can be selected. It's similar when on PC a you use your mouse to select a text. The whole mechanism (click and select the text as you drag your finger/mouse) is implemented in `Text area` and the Label widget only allows manual text selection with `lv_label_get_text_selection_start(label, start_char_index)` and `lv_label_get_text_selection_end(label, end_char_index)`.

如果通过 `LV_LABEL_TEXT_SELECTION` 启用, 可以选择部分文本。这与在 PC 上使用鼠标选择文本时类似。整个机制 (在拖动手指/鼠标时单击并选择文本) 在 `Text area` 中实现, 而标签小部件仅允许手动选择文本 `lv_label_get_text_selection_start(label, start_char_index)` 和 `lv_label_get_text_selection_end(label, end_char_index)`。

Very long texts (非常长的文本)

LVGL can efficiently handle very long (e.g. > 40k characters) labels by saving some extra data (~12 bytes) to speed up drawing. To enable this feature, set `LV_LABEL_LONG_TXT_HINT 1` in `lv_conf.h`.

LVGL 可以通过保存一些额外的数据 (~12 字节) 来有效地处理很长 (例如 > 40k 个字符) 的标签以加快绘图速度。要启用此功能, 请在 “`lv_conf.h`” 中设置 “`LV_LABEL_LONG_TXT_HINT 1`”。

Symbols (符号)

The labels can display symbols alongside letters (or on their own). Read the [Font](#) section to learn more about the symbols. 标签可以在字母旁边显示符号 (或单独显示)。阅读[字体](#)部分以了解有关符号的更多信息。

Events (事件)

No special events are sent by the Label.

See the events of the [Base object](#) too.

Learn more about [Events](#).

标签不发送特殊事件。

参见[Base object](#) 的事件。

详细了解[事件](#)。

Keys (按键)

No [Keys](#) are processed by the object type.

Learn more about [Keys](#).

对象类型不处理 [Keys](#)。

了解有关[键](#)的更多信息。

Example

C

Line wrap, recoloring and scrolling

```
#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_scr_act());
    lv_label_set_long_mode(label1, LV_LABEL_LONG_WRAP);      /*Break the long lines*/
}
```

(下页继续)

(续上页)

```

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 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_scr_act());
lv_label_set_long_mode(label2, LV_LABEL_LONG_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

```

```

label1 = lv.label(lv.scr_act())
label1.set_long_mode(lv.label.LONG.BREAK)      # Break the long lines
label1.set_recolor(True)                      # Enable re-coloring by commands in the text
label1.set_align(lv.label.ALIGN.CENTER)        # Center aligned lines
label1.set_text("#000080 Re-color# #0000ff words# #6666ff of a# label "
    "and wrap long text automatically.")
label1.set_width(150)
label1.align(None, lv.ALIGN.CENTER, 0, -30)

label2 = lv.label(lv.scr_act())
label2.set_long_mode(lv.label.LONG.SROLL_CIRC)  # Circular scroll
label2.set_width(150)
label2.set_text("It is a circularly scrolling text. ")
label2.align(None, lv.ALIGN.CENTER, 0, 30)

```

Text shadow

```

#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_scr_act());

```

(下页继续)

(续上页)

```

lv_obj_add_style(shadow_label, &style_shadow, 0);

/*Create the main label*/
lv_obj_t * main_label = lv_label_create(lv_scr_act());
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

```

```

# Create a style for the shadow
label_style = lv.style_t()
lv.style_copy(label_style, lv.style_plain)
label_style.text.opa = lv.OPA._50

# Create a label for the shadow first (it's in the background)
shadow_label = lv.label(lv.scr_act())
shadow_label.set_style(lv.label.STYLE.MAIN, label_style)

# Create the main label
main_label = lv.label(lv.scr_act())
main_label.set_text("A simple method to create\n" +
                    "shadows on text\n" +
                    "It even works with\n" +
                    "newlines      and spaces.")

# Set the same text for the shadow label
shadow_label.set_text(main_label.get_text())

# Position the main label
main_label.align(None, lv.ALIGN.CENTER, 0, 0)

# Shift the second label down and to the right by 1 pixel
shadow_label.align(main_label, lv.ALIGN.IN_TOP_LEFT, 1, 1)

```

MicroPython

No examples yet.

API

Typedefs

`typedef uint8_t lv_label_long_mode_t`

Enums

enum [anonymous]

Long mode behaviors. Used in 'lv_label_ext_t'

Values:

enumerator **LV_LABEL_LONG_WRAP**

Keep the object width, wrap the too long lines and expand the object height

enumerator **LV_LABEL_LONG_DOT**

Keep the size and write dots at the end if the text is too long

enumerator **LV_LABEL_LONG_SCROLL**

Keep the size and roll the text back and forth

enumerator **LV_LABEL_LONG_SCROLL_CIRCULAR**

Keep the size and roll the text circularly

enumerator **LV_LABEL_LONG_CLIP**

Keep the size and clip the text out of it

Functions

LV_EXPORT_CONST_INT(LV_LABEL_DOT_NUM)

LV_EXPORT_CONST_INT(LV_LABEL_POS_LAST)

LV_EXPORT_CONST_INT(LV_LABEL_TEXT_SELECTION_OFF)

*lv_obj_t *lv_label_create(lv_obj_t *parent)*

Create a label objects

参数 parent -- pointer to an object, it will be the parent of the new label.

返回 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.

参数

- **label** -- 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, ...)`

`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 exist.

参数

- **label** -- 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 longer text then the object size

参数

- **label** -- 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_recolor(lv_obj_t *obj, bool en)`

`void lv_label_set_text_sel_start(lv_obj_t *obj, uint32_t index)`

Set where text selection should start

参数

- **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_sel_end(lv_obj_t *obj, uint32_t index)`

Set where text selection should end

参数

- **obj** -- pointer to a label object
- **index** -- character index where selection should end. LV_LABEL_TEXT_SELECTION_OFF for no selection

`char *lv_label_get_text(const lv_obj_t *obj)`

Get the text of a label

参数 **obj** -- pointer to a label object

返回 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

参数 **obj** -- pointer to a label object

返回 the current long mode

`bool lv_label_get_recolor(const lv_obj_t *obj)`

Get the recoloring attribute

参数 **obj** -- pointer to a label object

返回 true: recoloring is enabled, false: disable

`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

参数

- **obj** -- pointer to a label object
- **index** -- 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)`

Get the index of letter on a relative point of a label.

参数

- **obj** -- pointer to label object
- **pos** -- pointer to point with coordinates on a label

返回 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.

参数

- **label** -- Label object
- **pos** -- Point to check for character under

返回 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.

参数 **obj** -- pointer to a label object.

返回 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.

参数 **obj** -- pointer to a label object.

返回 selection end index. LV_LABEL_TXT_SEL_OFF if nothing is selected.

`void lv_label_ins_text(lv_obj_t *obj, uint32_t pos, const char *txt)`

Insert a text to a label. The label text can not be static.

参数

- **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 can not be static.

参数

- **label** -- 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
struct lv_label_t
```

Public Members

```
lv_obj_t obj
char *text
char *tmp_ptr
char tmp[LV_LABEL_DOT_NUM + 1]
union lv_label_t::[anonymous] dot
uint32_t dot_end
lv_draw_label_hint_t hint
uint32_t sel_start
uint32_t sel_end
lv_point_t offset
lv_label_long_mode_t long_mode
uint8_t static_txt
uint8_t recolor
uint8_t expand
uint8_t dot_tmp_alloc
```

6.2.10 Line (线) (lv_line)

Overview (概述)

The Line object is capable of drawing straight lines between a set of points.

Line 对象能够在一组点之间绘制直线。

Parts and Styles (零件和样式)

- LV_PART_MAIN uses all the typical background properties and line style properties.
- LV_PART_MAIN 使用所有典型的背景属性和线条样式属性。

Usage (用法)

Set points (设置点)

The points have to be stored in an `lv_point_t` array and passed to the object by the `lv_line_set_points(lines, point_array, point_cnt)` function.

点必须存储在一个 `lv_point_t` 数组中，并通过 `lv_line_set_points(lines, point_array, point_cnt)` 函数传递给对象。

Auto-size (自动调整大小)

By default the Line's width and height are set to `LV_SIZE_CONTENT`. This means it will automatically set its size to fit all the points. If the size is set explicitly, parts on the line may not be visible.

默认情况下，Line 的宽度和高度被设置为 `LV_SIZE_CONTENT`。这意味着它将自动设置其大小以适应所有的点。如果明确设置了尺寸，线上的部分可能不可见。

Invert y (反转 y)

By default, the `y == 0` point is in the top of the object. It might be counter-intuitive in some cases so the y coordinates can be inverted with `lv_line_set_y_invert(line, true)`. In this case, `y == 0` will be the bottom of the object. `y invert` is disabled by default.

默认情况下，`y == 0` 点是在物体的顶部。这在某些情况下可能是不直观的，所以可以用 `lv_line_set_y_invert(line, true)` 来反转 y 坐标。在这种情况下，`y == 0` 将是物体的底部。默认情况下，`y invert` 是禁用的。

Events (事件)

Only the [Generic events](#) are sent by the object type.

See the events of the [Base object](#) too.

Learn more about [Events](#).

对象类型仅发送 [通用事件](#)。

参见[Base object](#) 的事件。

详细了解事件。

Keys (按键)

No [Keys](#) are processed by the object type.

Learn more about [Keys](#).

对象类型不处理 [Keys](#)。

了解有关[键](#)的更多信息。

Example

C

Simple Line

```
#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_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_scr_act());
    lv_line_set_points(line1, line_points, 5);      /*Set the points*/
    lv_obj_add_style(line1, &style_line, 0);
    lv_obj_center(line1);
}

#endif
```

```

# Create an array for the points of the line
line_points = [ {"x":5, "y":5},
                {"x":70, "y":70},
                {"x":120, "y":10},
                {"x":180, "y":60},
                {"x":240, "y":10}]

# Create new style (thick dark blue)
style_line = lv.style_t()
lv.style_copy(style_line, lv.style_plain)
style_line.line.color = lv.color_make(0x00, 0x3b, 0x75)
style_line.line.width = 3
style_line.line.rounded = 1

# Copy the previous line and apply the new style
line1 = lv.line(lv.scr_act())
line1.set_points(line_points, len(line_points))      # Set the points
line1.set_style(lv.line.STYLE.MAIN, style_line)
line1.align(None, lv.ALIGN.CENTER, 0, 0)

```

MicroPython

No examples yet.

API

Functions

`lv_obj_t *lv_line_create(lv_obj_t *parent)`

Create a line objects

参数 **par** -- pointer to an object, it will be the parent of the new line

返回 pointer to the created line

`void lv_line_set_points(lv_obj_t *obj, const lv_point_t points[], uint16_t point_num)`

Set an array of points. The line object will connect these points.

参数

- **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_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.

参数

- **obj** -- pointer to a line object
- **en** -- true: enable the y inversion, false:disable the y inversion

`bool lv_line_get_y_invert(const lv_obj_t *obj)`

Get the y inversion attribute

参数 **obj** -- pointer to a line object

返回 true: y inversion is enabled, false: disabled

Variables

const lv_obj_class_t **lv_line_class**

struct **lv_line_t**

Public Members

lv_obj_t **obj**

const lv_point_t ***point_array**

Pointer to an array with the points of the line

uint16_t **point_num**

Number of points in 'point_array'

uint8_t **y_inv**

1: y == 0 will be on the bottom

6.2.11 Roller (滚轮) (**lv_roller**)

Overview (概述)

Roller allows you to simply select one option from a list by scrolling.

滚轮允许您通过滚动从列表中简单地选择一个选项。

Parts and Styles (零件和样式)

- **LV_PART_MAIN** The background of the roller uses all the typical background properties and text style properties. **style_text_line_space** adjusts the space between the options. When the Roller is scrolled and doesn't stop exactly on an option it will scroll to the nearest valid option automatically in **anim_time** milliseconds as specified in the style.
- **LV_PART_SELECTED** The selected option in the middle. Besides the typical background properties it uses the text style properties to change the appearance of the text in the selected area.
- **LV_PART_MAIN** 滚轮的背景使用了所有典型的背景属性和文本样式属性。**style_text_line_space** 调整选项之间的空间。当滚轮滚动并且没有完全停在一个选项上时，它将按照样式中指定的 **anim_time** 毫秒自动滚动到最近的有效选项。
- **LV_PART_SELECTED** 中间选中的选项。除了典型的背景属性之外，它还使用文本样式属性来更改所选区域中文本的外观。

Usage (用法)

Set options (设置点)

Options are passed to the Roller as a string with `lv_roller_set_options(roller, options, LV_ROLLER_MODE_NORMAL/INFINITE)`. The options should be separated by \n. For example: "First\nSecond\nThird".

`LV_ROLLER_MODE_INFINITE` makes the roller circular.

You can select an option manually with `lv_roller_set_selected(roller, id, LV_ANIM_ON/OFF)`, where *id* is the index of an option.

选项作为带有 `lv_roller_set_options(roller, options, LV_ROLLER_MODE_NORMAL/INFINITE)` 的字符串传递给 Roller。选项应该用 \n 分隔。例如: "First\nSecond\nThird"。

`LV_ROLLER_MODE_INFINITE` 使滚轮呈圆形。

您可以使用 `lv_roller_set_selected(roller, id, LV_ANIM_ON/OFF)` 手动选择选项，其中 *id* 是选项的索引。

Get selected option (获取选中的选项)

The get the *index* of the currently selected option use `lv_roller_get_selected(roller)`.

`lv_roller_get_selected_str(roller, buf, buf_size)` will copy the name of the selected option to *buf*.

获取当前选定选项的 *index* 使用 `lv_roller_get_selected(roller)`。

`lv_roller_get_selected_str(roller, buf, buf_size)` 会将所选项的名称复制到 *buf*。

Visible rows (可见行)

The number of visible rows can be adjusted with `lv_roller_set_visible_row_count(roller, num)`.

This function calculates the height with the current style. If the font, line space, border width, etc of the roller changes this function needs to be called again.

可见行数可以通过 `lv_roller_set_visible_row_count(roller, num)` 进行调整。

此函数计算当前样式的高度。如果滚轮的字体、行距、边框宽度等发生变化，则需要再次调用此函数。

Events (事件)

- `LV_EVENT_VALUE_CHANGED` Sent when a new option is selected.

See the events of the *Base object* too.

Learn more about *Events*.

- `LV_EVENT_VALUE_CHANGED` 选择新选项时发送。

参见*Base object* 的事件。

详细了解事件。

Keys (按键)

- LV_KEY_RIGHT/DOWN Select the next option
- LV_KEY_LEFT/UP Select the previous option
- LY_KEY_ENTER Apply the selected option (Send LV_EVENT_VALUE_CHANGED event)
- LV_KEY_RIGHT/DOWN 选择下一个选项
- LV_KEY_LEFT/UP 选择上一个选项
- LY_KEY_ENTER 应用选择的选项 (发送 LV_EVENT_VALUE_CHANGED 事件)

Example

C

Simple Roller

```
#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(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_scr_act());
    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);
}
```

(下页继续)

(续上页)

```

    lv_obj_add_event_cb(roller1, event_handler, LV_EVENT_ALL, NULL);
}

#endif

```

```

def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
        option = " "*10
        obj.get_selected_str(option, len(option))
        print("Selected month: %s" % option.strip())

roller1 = lv.roller(lv.scr_act())
roller1.set_options("\n".join([
    "January",
    "February",
    "March",
    "April",
    "May",
    "June",
    "July",
    "August",
    "September",
    "October",
    "November",
    "December"]), lv.roller.MODE.INFINITE)

roller1.set_visible_row_count(4)
roller1.align(None, lv.ALIGN.CENTER, 0, 0)
roller1.set_event_cb(event_handler)

```

Styling the roller

```

#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(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);
}

```

(下页继续)

(续上页)

```

lv_style_set_text_font(&style_sel, &lv_font_montserrat_22);

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_scr_act());
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_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_scr_act());
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_scr_act());
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

```

Error encountered **while** trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/roller/lv_example_roller_2.py

MicroPython

No examples yet.

API

TypeDefs

`typedef uint8_t lv_roller_mode_t`

Enums

enum [anonymous]

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).

Functions

`lv_obj_t *lv_roller_create(lv_obj_t *parent)`

Create a roller objects

参数 **parent** -- pointer to an object, it will be the parent of the new roller.

返回 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

参数

- **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, uint16_t sel_opt, lv_anim_enable_t anim)`

Set the selected option

参数

- **obj** -- pointer to a roller object
- **sel_opt** -- index of the selected option (0 ... number of option - 1);
- **anim_en** -- LV_ANIM_ON: set with animation; LV_ANOM_OFF set immediately

`void lv_roller_set_visible_row_count(lv_obj_t *obj, uint8_t row_cnt)`

Set the height to show the given number of rows (options)

参数

- **obj** -- pointer to a roller object

- **row_cnt** -- number of desired visible rows

`uint16_t lv_roller_get_selected(const lv_obj_t *obj)`
Get the index of the selected option

参数 **obj** -- pointer to a roller object

返回 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.

参数

- **obj** -- pointer to dlist 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

参数 **obj** -- pointer to roller object

返回

the options separated by '

'-s (E.g. "Option1\nOption2\nOption3")

`uint16_t lv_roller_get_option_cnt(const lv_obj_t *obj)`
Get the total number of options

参数 **obj** -- pointer to a roller object

返回 the total number of options

Variables

`const lv_obj_class_t lv_roller_class`
`struct lv_roller_t`

Public Members

lv_obj_t **obj**

`uint16_t option_cnt`
Number of options

`uint16_t sel_opt_id`
Index of the current option

`uint16_t sel_opt_id_ori`
Store the original index on focus

lv_roller_mode_t **mode**

`uint32_t moved`

6.2.12 Slider (滑杆) (lv_slider)

Overview (概述)

The Slider object looks like a *Bar* supplemented with a knob. The knob can be dragged to set a value. Just like Bar, Slider can be vertical or horizontal.

Slider 对象看起来像一个 *Bar* 补充了一个旋钮。可以拖动旋钮来设置一个值。就像 Bar 一样，Slider 可以是垂直的或水平的。

Parts and Styles (部件和样式)

- **LV_PART_MAIN** The background of the slider. Uses all the typical background style properties. **padding** makes the indicator smaller in the respective direction.
- **LV_PART_INDICATOR** The indicator that shows the current state of the slider. Also uses all the typical background style properties.
- **LV_PART_KNOB** A rectangle (or circle) drawn at the current value. Also uses all the typical background properties to describe the knob(s). By default the knob is square (with a optional corner radius) with side length equal to the smaller side of the slider. The knob can be made larger with the **padding** values. Padding values can be asymmetric too.
- **LV_PART_MAIN** 滑块的背景。使用所有典型的背景样式属性。**padding** 使指标在相应方向上变小。
- **LV_PART_INDICATOR** 显示滑块当前状态的指示器。还使用所有典型的背景样式属性。
- **LV_PART_KNOB** 在当前值处绘制的矩形（或圆形）。还使用所有典型的背景属性来描述旋钮。默认情况下，旋钮是方形的（带有可选的圆角半径），边长等于滑块的较小边。可以使用“padding”值使旋钮变大。填充值也可以是不对称的。

Usage (用法)

Value and range (值和范围)

To set an initial value use `lv_slider_set_value(slider, new_value, LV_ANIM_ON/OFF)`. The animation time is set by the styles' `anim_time` property.

To specify the range (min, max values), `lv_slider_set_range(slider, min , max)` can be used.

要设置初始值，请使用 `lv_slider_set_value(slider, new_value, LV_ANIM_ON/OFF)`。动画时间由样式的 `anim_time` 属性设置。

要指定范围（最小值、最大值），可以使用 `lv_slider_set_range(slider, min , max)`。

Modes (模式)

The slider can be one the following modes:

- **LV_SLIDER_MODE_NORMAL** A normal slider as described above
- **LV_SLIDER_SYMMETRICAL** Draw the indicator from the zero value to current value. Requires negative minimum range and positive maximum range.
- **LV_SLIDER_RANGE** Allows setting the start value too by `lv_bar_set_start_value(bar, new_value, LV_ANIM_ON/OFF)`. The start value has to be always smaller than the end value.

The mode can be changed with `lv_slider_set_mode(slider, LV_SLIDER_MODE_...)`

滑块可以是以下模式之一：

- `LV_SLIDER_MODE_NORMAL` 如上所述的普通滑块
- `LV_SLIDER_MODE_SYMMETRICAL` 从零值到当前值绘制指标。需要负最小范围和正最大范围。
- `LV_SLIDER_MODE_RANGE` 也允许通过 `lv_bar_set_start_value(bar, new_value, LV_ANIM_ON/OFF)` 设置起始值。起始值必须始终小于结束值。

可以使用 `lv_slider_set_mode(slider, LV_SLIDER_MODE_...)` 更改模式

Knob-only mode (仅旋钮模式)

Normally, the slider can be adjusted either by dragging the knob, or by clicking on the slider bar. In the latter case the knob moves to the point clicked and slider value changes accordingly. In some cases it is desirable to set the slider to react on dragging the knob only. This feature is enabled by adding the `LV_OBJ_FLAG_ADV_HITTEST: lv_obj_add_flag(slider, LV_OBJ_FLAG_ADV_HITTEST)`.

通常，可以通过拖动旋钮或单击滑块条来调整滑块。在后一种情况下，旋钮会移动到单击的点，滑块值会相应更改。在某些情况下，需要将滑块设置为仅对拖动旋钮做出反应。通过添加 `LV_OBJ_FLAG_ADV_HITTEST: lv_obj_add_flag(slider, LV_OBJ_FLAG_ADV_HITTEST)` 来启用此功能。

Events (事件)

- `LV_EVENT_VALUE_CHANGED` Sent while the slider is being dragged or changed with keys. The event is sent continuously while the slider is dragged and once when released. Use `lv_slider_is_dragged` to determine whether the Slider is still being dragged or has just been released.
- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` are sent for the following parts.
 - `LV_SLIDER_DRAW_PART_KNOB` The main (right) knob of the slider
 - * `part: LV_PART_KNOB`
 - * `draw_area`: area of the indicator
 - * `rect_dsc`
 - * `id: 0`
 - `LV_SLIDER_DRAW_PART_KNOB` The left knob of the slider
 - * `part: LV_PART_KNOB`
 - * `draw_area`: area of the indicator
 - * `rect_dsc`
 - * `id: 1`

See the events of the [Bar](#) too.

Learn more about [Events](#).

- `LV_EVENT_VALUE_CHANGED` 在滑块被拖动或使用键更改时发送。拖动滑块时连续发送事件，释放时发送一次。使用 `lv_slider_is_dragged` 来确定 Slider 是仍在被拖动还是刚刚被释放。
- `LV_EVENT_DRAW_PART_BEGIN` 和 `LV_EVENT_DRAW_PART_END` 被发送用于以下部分。
 - `LV_SLIDER_DRAW_PART_KNOB` 滑块的主（右）旋钮
 - * 部分： `LV_PART_KNOB`

- * `draw_area`: 指标的区域 - `rect_dsc.id`: 0
- `LV_SLIDER_DRAW_PART_KNOB` 滑块的左侧旋钮
 - * 部分: `LV_PART_KNOB`
 - * `draw_area`: 指标的区域 - `rect_dsc.id`: 1

也可以查看 [Bar](#) 的事件。

详细了解 [事件](#)。

Keys (按键)

- `LV_KEY_UP/RIGHT` Increment the slider's value by 1
- `LV_KEY_DOWN/LEFT` Decrement the slider's value by 1

Learn more about [Keys](#).

- `LV_KEY_UP/RIGHT` 将滑块的值增加 1
- `LV_KEY_DOWN/LEFT` 将滑块的值减 1

了解有关[键](#)的更多信息。

Example

C

Simple Slider

```
#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_scr_act());
    lv_obj_center(slider);
    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

    /*Create a label below the slider*/
    slider_label = lv_label_create(lv_scr_act());
    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(e);
```

(下页继续)

(续上页)

```

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

```

```

def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
        print("Value: %d" % obj.get_value())

# Create styles
style_bg = lv.style_t()
style_indic = lv.style_t()
style_knob = lv.style_t()

lv.style_copy(style_bg, lv.style_pretty)
style_bg.body.main_color = lv.color_make(0,0,0)
style_bg.body.grad_color = lv.color_make(0x80, 0x80, 0x80)
style_bg.body.radius = 800 # large enough to make a circle
style_bg.body.border.color = lv.color_make(0xff,0xff,0xff)

lv.style_copy(style_indic, lv.style_pretty_color)
style_indic.body.radius = 800
style_indic.body.shadow.width = 8
style_indic.body.shadow.color = style_indic.body.main_color
style_indic.body.padding.left = 3
style_indic.body.padding.right = 3
style_indic.body.padding.top = 3
style_indic.body.padding.bottom = 3

lv.style_copy(style_knob, lv.style_pretty)
style_knob.body.radius = 800
style_knob.body.opa = lv.OPA._70
style_knob.body.padding.top = 10
style_knob.body.padding.bottom = 10

# Create a slider
slider = lv.slider(lv.scr_act())
slider.set_style(lv.slider.STYLE.BG, style_bg)
slider.set_style(lv.slider.STYLE.INDIC, style_indic)
slider.set_style(lv.slider.STYLE.KNOB, style_knob)
slider.align(None, lv.ALIGN.CENTER, 0, 0)
slider.set_event_cb(event_handler)

```

Slider with custom style

```
#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_scr_act());
    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);
}
```

(下页继续)

(续上页)

```

    lv_obj_add_style(slider, &style_pressed_color, LV_PART_KNOB | LV_STATE_PRESSED);

    lv_obj_center(slider);
}

#endif

```

```

def slider_event_cb(slider, event):
    if event == lv.EVENT.VALUE_CHANGED:
        slider_label.set_text("%u" % slider.get_value())

# Create a slider in the center of the display
slider = lv.slider(lv.scr_act())
slider.set_width(200)
slider.align(None, lv.ALIGN.CENTER, 0, 0)
slider.set_event_cb(slider_event_cb)
slider.set_range(0, 100)

# Create a label below the slider
slider_label = lv.label(lv.scr_act())
slider_label.set_text("0")
slider_label.set_auto_realign(True)
slider_label.align(slider, lv.ALIGN.OUT_BOTTOM_MID, 0, 10)

# Create an informative label
info = lv.label(lv.scr_act())
info.set_text("""Welcome to the slider+label demo!
Move the slider and see that the label
updates to match it.""")
info.align(None, lv.ALIGN.IN_TOP_LEFT, 10, 10)

```

Slider with extended drawer

```

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

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_scr_act());
    lv_obj_center(slider);

    lv_slider_set_mode(slider, LV_SLIDER_MODE_RANGE);
    lv_slider_set_value(slider, 70, LV_ANIM_OFF);
    lv_slider_set_left_value(slider, 20, LV_ANIM_OFF);

    lv_obj_add_event_cb(slider, slider_event_cb, LV_EVENT_ALL, NULL);
}

```

(下页继续)

(续上页)

```

    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(e);

    /*Provide some extra space for the value*/
    if(code == LV_EVENT_REFR_EXT_DRAW_SIZE) {
        lv_coord_t * size = lv_event_get_param(e);
        *size = LV_MAX(*size, 50);
    }
    else if(code == LV_EVENT_DRAW_PART_END) {
        lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
        if(dsc->part == LV_PART_INDICATOR) {
            char buf[16];
            lv_snprintf(buf, sizeof(buf), "%d - %d", lv_slider_get_left_value(obj),
            ↪lv_slider_get_value(obj));

            lv_point_t label_size;
            lv_txt_get_size(&label_size, buf, LV_FONT_DEFAULT, 0, 0, LV_COORD_MAX, 0);
            lv_area_t label_area;
            label_area.x1 = dsc->draw_area->x1 + lv_area_get_width(dsc->draw_area) / 2 -
            ↪2 - label_size.x / 2;
            label_area.x2 = label_area.x1 + label_size.x;
            label_area.y2 = dsc->draw_area->y1 - 10;
            label_area.y1 = label_area.y2 - label_size.y;

            lv_draw_label_dsc_t label_draw_dsc;
            lv_draw_label_dsc_init(&label_draw_dsc);

            lv_draw_label(&label_area, dsc->clip_area, &label_draw_dsc, buf, NULL);
        }
    }
}

#endif

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/slider/lv_example_slider_3.py

MicroPython

No examples yet.

API

Typedefs

`typedef uint8_t lv_slider_mode_t`

Enums

enum [anonymous]

Values:

- enumerator `LV_SLIDER_MODE_NORMAL`
- enumerator `LV_SLIDER_MODE_SYMMETRICAL`
- enumerator `LV_SLIDER_MODE_RANGE`

Functions

`lv_obj_t *lv_slider_create(lv_obj_t *parent)`

Create a slider objects

参数 `parent` -- pointer to an object, it will be the parent of the new slider.

返回 pointer to the created slider

`static inline void lv_slider_set_value(lv_obj_t *obj, int32_t value, lv_anim_enable_t anim)`

Set a new value on the slider

参数

- `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

`static inline void lv_slider_set_left_value(lv_obj_t *obj, int32_t value, lv_anim_enable_t anim)`

Set a new value for the left knob of a slider

参数

- `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

`static inline void lv_slider_set_range(lv_obj_t *obj, int32_t min, int32_t max)`

Set minimum and the maximum values of a bar

参数

- `obj` -- pointer to the slider object
- `min` -- minimum value
- `max` -- maximum value

static inline void **lv_slider_set_mode**(*lv_obj_t* *obj, *lv_slider_mode_t* mode)
Set the mode of slider.

参数

- **obj** -- pointer to a slider object
- **mode** -- the mode of the slider. See ::lv_slider_mode_t

static inline int32_t **lv_slider_get_value**(const *lv_obj_t* *obj)
Get the value of the main knob of a slider

参数 obj -- pointer to a slider object**返回** the value of the main knob of the slider

static inline int32_t **lv_slider_get_left_value**(const *lv_obj_t* *obj)
Get the value of the left knob of a slider

参数 obj -- pointer to a slider object**返回** the value of the left knob of the slider

static inline int32_t **lv_slider_get_min_value**(const *lv_obj_t* *obj)
Get the minimum value of a slider

参数 obj -- pointer to a slider object**返回** the minimum value of the slider

static inline int32_t **lv_slider_get_max_value**(const *lv_obj_t* *obj)
Get the maximum value of a slider

参数 obj -- pointer to a slider object**返回** the maximum value of the slider

bool **lv_slider_is_dragged**(const *lv_obj_t* *obj)
Give the slider is being dragged or not

参数 obj -- pointer to a slider object**返回** true: drag in progress false: not dragged

static inline *lv_slider_mode_t* **lv_slider_get_mode**(*lv_obj_t* *slider)
Get the mode of the slider.

参数 obj -- pointer to a bar object**返回** see ::lv_slider_mode_t**Variables**

const *lv_obj_class_t* **lv_slider_class**

struct **lv_slider_t**

Public Members

```
lv_bar_t bar
lv_area_t left_knob_area
lv_area_t right_knob_area
int32_t *value_to_set
uint8_t dragging
uint8_t left_knob_focus
```

6.2.13 Switch (开关) (lv_switch)

Overview (概述)

The Switch looks like a little slider and can be used to turn something on and off.

开关看起来像一个小滑块，可用于打开和关闭某些东西。

Parts and Styles (部件和样式)

- **LV_PART_MAIN** The background of the switch uses all the typical background style properties. **padding** makes the indicator smaller in the respective direction.
- **LV_PART_INDICATOR** The indicator that shows the current state of the switch. Also uses all the typical background style properties.
- **LV_PART_KNOB** A rectangle (or circle) drawn at left or right side of the indicator. Also uses all the typical background properties to describe the knob(s). By default the knob is square (with a optional corner radius) with side length equal to the smaller side of the slider. The knob can be made larger with the **padding** values. Padding values can be asymmetric too.
- **LV_PART_MAIN** 开关的背景使用所有典型的背景样式属性。**padding** 使指标在相应方向上变小。
- **LV_PART_INDICATOR** 显示开关当前状态的指示器。还使用所有典型的背景样式属性。
- **LV_PART_KNOB** 在指标左侧或右侧绘制的矩形 (或圆形)。还使用所有典型的背景属性来描述旋钮。默认情况下，旋钮是方形的 (带有可选的圆角半径)，边长等于滑块的较小边。可以使用“padding”值使旋钮变大。填充值也可以是不对称的。

Usage (用法)

Change state (改变状态)

When the switch is turned on it goes to **LV_STATE_CHECKED**. To get the current state of the switch use `lv_obj_has_state(switch, LV_STATE_CHECKED)`. To manually turn the switch on/off call `lvobj_add/clear_state(switch, LV_STATE_CHECKED)`.

当开关打开时，它会进入“**LV_STATE_CHECKED**”。要获取开关的当前状态，请使用 `lv_obj_has_state(switch, LV_STATE_CHECKED)`。要手动打开/关闭开关，请调用 `lvobj_add/clear_state(switch, LV_STATE_CHECKED)`。

Events (事件)

- LV_EVENT_VALUE_CHANGED Sent when the switch changes state.

See the events of the [Base object](#) too.

Learn more about [Events](#).

- LV_EVENT_VALUE_CHANGED 当开关改变状态时发送。

参见[Base object](#) 的事件。

详细了解[事件](#)。

Keys (按键)

- LV_KEY_UP/RIGHT Turns on the slider
- LV_KEY_DOWN/LEFT Turns off the slider
- LV_KEY_ENTER Toggles the switch

Learn more about [Keys](#).

- LV_KEY_UP/RIGHT 打开滑块
- LV_KEY_DOWN/LEFT 关闭滑块
- LV_KEY_ENTER 切换开关

了解有关[键](#)的更多信息。

Example

C

Simple Switch

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↳ lvgl_docs_8.x/examples/widgets/lv_example_switch/lv_example_switch_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↳ lvgl_docs_8.x/examples/widgets/lv_example_switch/lv_example_switch_1.py
```

MicroPython

No examples yet.

API

Functions

`lv_obj_t *lv_switch_create(lv_obj_t *parent)`

Create a switch objects

参数 **parent** -- pointer to an object, it will be the parent of the new switch

返回 pointer to the created switch

Variables

```
const lv_obj_class_t lv_switch_class
struct lv_switch_t
```

Public Members

`lv_obj_t obj`

6.2.14 Table (表) (lv_table)

Overview

Tables, as usual, are built from rows, columns, and cells containing texts.

The Table object is very lightweight because only the texts are stored. No real objects are created for cells but they are just drawn on the fly.

表格是由包含文本的行、列和单元格构建的。表格对象非常轻量级，因为仅存储文本。没有为细胞创建真实的对象，但它们只是即时绘制的。

Parts and Styles (部件和样式)

- **LV_PART_MAIN** The background of the table uses all the typical background style properties.
- **LV_PART_ITEMS** The cells of the table also use all the typical background style properties and the text properties.
- **LV_PART_MAIN** 表格的背景使用了所有典型的背景样式属性。
- **LV_PART_ITEMS** 表格的单元格也使用所有典型的背景样式属性和文本属性。

Usage (用法)

Set cell value (设置单元格的值)

The cells can store only text so numbers need to be converted to text before displaying them in a table.

`lv_table_set_cell_value(table, row, col, "Content")`. The text is saved by the table so it can be even a local variable.

Line breaks can be used in the text like "`Value\n60.3`".

New rows and columns are automatically added if required

单元格只能存储文本，因此在将数字显示在表格中之前，需要将其转换为文本。

`lv_table_set_cell_value(table, row, col, "Content")`。文本由表保存，因此它甚至可以是局部变量。

可以在文本中使用换行符，例如 "`Value\n60.3`"。

需要自动添加新的行和列

Rows and Columns (行和列)

To explicitly set number of rows and columns use `lv_table_set_row_cnt(table, row_cnt)` and `lv_table_set_col_cnt(table, col_cnt)`

要明确设置行数和列数，请使用 `lv_table_set_row_cnt(table, row_cnt)` 和 `lv_table_set_col_cnt(table, col_cnt)`

Width and Height (宽度和高度)

The width of the columns can be set with `lv_table_set_col_width(table, col_id, width)`. The overall width of the Table object will be set to the sum of columns widths.

The height is calculated automatically from the cell styles (font, padding etc) and the number of rows.

列的宽度可以通过 `lv_table_set_col_width(table, col_id, width)` 设置。Table 对象的总宽度将设置为列宽的总和。

高度是根据单元格样式（字体、填充等）和行数自动计算的。

Merge cells (合并单元格)

Cells can be merged horizontally with `lv_table_set_cell_merge_right(table, col, row, true)`. To merge more adjacent cells call this function for each cell.

单元格可以使用 `lv_table_set_cell_merge_right(table, col, row, true)` 进行水平合并。要合并更多相邻单元格，请为每个单元格调用此函数。

Scroll (滚动)

If the label's width or height is set to `LV_SIZE_CONTENT` that size will be used to show the whole table in the respective direction. E.g. `lv_obj_set_size(table, LV_SIZE_CONTENT, LV_SIZE_CONTENT)` automatically sets the table size to show all the columns and rows.

If the width or height is set to a smaller number than the "intrinsic" size then the table becomes scrollable.

如果标签的宽度或高度设置为“`LV_SIZE_CONTENT`”，则该尺寸将用于在相应方向上显示整个表格。例如。`lv_obj_set_size(table, LV_SIZE_CONTENT, LV_SIZE_CONTENT)` 自动设置表格大小以显示所有列和行。

如果宽度或高度设置为小于“固有”大小的数字，则表格变为可滚动的。

Events (事件)

- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` are sent for the following types:
 - `LV_TABLE_DRAW_PART_CELL` The individual cells of the table
 - * `part`: `LV_PART_ITEMS`
 - * `draw_area`: area of the indicator
 - * `rect_dsc`
 - * `label_dsc`
 - * `id`: current row × col count + current column

See the events of the *Base object* too.

Learn more about *Events*.

- 为以下类型发送 `LV_EVENT_DRAW_PART_BEGIN` 和 `LV_EVENT_DRAW_PART_END`:
 - `LV_TABLE_DRAW_PART_CELL` 表格的各个单元格
 - * 部分: `LV_PART_ITEMS`
 - * `draw_area`: 指标的区域 - `rect_dsc`
 - * `label_dsc`
 - * `id`: 当前行 × 列数 + 当前列

参见*Base object* 的事件。

详细了解事件。

Keys (按键)

No *Keys* are processed by the object type.

Learn more about *Keys*.

对象类型不处理 *Keys*。

了解有关键的更多信息。

Example

C

Simple table

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

static void draw_part_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_param(e);
    /*If the cells are drawn...*/
    if(dsc->part == LV_PART_ITEMS) {
        uint32_t row = dsc->id / lv_table_get_col_cnt(obj);
        uint32_t col = dsc->id - row * lv_table_get_col_cnt(obj);

        /*Make the texts in the first cell center aligned*/
        if(row == 0) {
            dsc->label_dsc->align = LV_TEXT_ALIGN_CENTER;
            dsc->rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_BLUE), ▶
            dsc->rect_dsc->bg_color, LV_OPA_20);
            dsc->rect_dsc->bg_opa = LV_OPA_COVER;
        }
        /*In the first column align the texts to the right*/
        else if(col == 0) {
            dsc->label_dsc->flag = LV_TEXT_ALIGN_RIGHT;
        }

        /*MAke every 2nd row grayish*/
        if((row != 0 && row % 2) == 0) {
            dsc->rect_dsc->bg_color = lv_color_mix(lv_palette_main(LV_PALETTE_GREY), ▶
            dsc->rect_dsc->bg_color, LV_OPA_10);
            dsc->rect_dsc->bg_opa = LV_OPA_COVER;
        }
    }
}

void lv_example_table_1(void)
{
    lv_obj_t * table = lv_table_create(lv_scr_act());

    /*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");
}
```

(下页继续)

(续上页)

```

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 apply some custom drawing*/
lv_obj_add_event_cb(table, draw_part_event_cb, LV_EVENT_DRAW_PART_BEGIN, NULL);
}

#endif

```

```

# Create a normal cell style
style_cell1 = lv.style_t()
lv.style_copy(style_cell1, lv.style_plain)
style_cell1.body.border.width = 1
style_cell1.body.border.color = lv.color_make(0,0,0)

# Create a header cell style
style_cell2 = lv.style_t()
lv.style_copy(style_cell2, lv.style_plain)
style_cell2.body.border.width = 1
style_cell2.body.border.color = lv.color_make(0,0,0)
style_cell2.body.main_color = lv.color_make(0xC0, 0xC0, 0xC0)
style_cell2.body.grad_color = lv.color_make(0xC0, 0xC0, 0xC0)

table = lv.table(lv.scr_act())
table.set_style(lv.table.STYLE.CELL1, style_cell1)
table.set_style(lv.table.STYLE.CELL2, style_cell2)
table.set_style(lv.table.STYLE.BG, lv.style_transp_tight)
table.set_col_cnt(2)
table.set_row_cnt(4)
table.align(None, lv.ALIGN.CENTER, 0, 0)

# Make the cells of the first row center aligned
table.set_cell_align(0, 0, lv.label.ALIGN.CENTER)
table.set_cell_align(0, 1, lv.label.ALIGN.CENTER)

# Make the cells of the first row TYPE = 2 (use `style_cell2`)
table.set_cell_type(0, 0, 2)
table.set_cell_type(0, 1, 2)

# Fill the first column
table.set_cell_value(0, 0, "Name")
table.set_cell_value(1, 0, "Apple")
table.set_cell_value(2, 0, "Banana")
table.set_cell_value(3, 0, "Citron")

# Fill the second column
table.set_cell_value(0, 1, "Price")

```

(下页继续)

(续上页)

```
table.set_cell_value(1, 1, "$7")
table.set_cell_value(2, 1, "$4")
table.set_cell_value(3, 1, "$6")
```

Lightweighted list from table

```
#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(e);
    lv_obj_draw_part_dsc_t * dsc = lv_event_get_draw_part_dsc(e);
    /*If the cells are drawn...*/
    if(dsc->part == LV_PART_ITEMS) {
        bool chk = lv_table_has_cell_ctrl(obj, dsc->id, 0, LV_TABLE_CELL_CTRL_CUSTOM_
→1);

        lv_rect_dsc_t rect_dsc;
        lv_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 = dsc->draw_area->x2 - 50;
        sw_area.x2 = sw_area.x1 + 40;
        sw_area.y1 = dsc->draw_area->y1 + lv_area_get_height(dsc->draw_area) / 2 - u
→10;
        sw_area.y2 = sw_area.y1 + 20;
        lv_rect(&sw_area, dsc->clip_area, &rect_dsc);

        rect_dsc.bg_color = lv_color_white();
        if(chk) {
            sw_area.x2 -= 2;
            sw_area.x1 = sw_area.x2 - 16;
        } else {
            sw_area.x1 += 2;
            sw_area.x2 = sw_area.x1 + 16;
        }
        sw_area.y1 += 2;
        sw_area.y2 -= 2;
        lv_rect(&sw_area, dsc->clip_area, &rect_dsc);
    }
}

static void change_event_cb(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    uint16_t col;
    uint16_t row;
    lv_table_get_selected_cell(obj, &row, &col);
```

(下页继续)

(续上页)

```

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_add_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_scr_act());

    /*Set a smaller height to the table. It'll make it scrollable*/
    lv_obj_set_size(table, 150, 200);

    lv_table_set_col_width(table, 0, 150);
    lv_table_set_row_cnt(table, ITEM_CNT); /*Not required but avoids a lot of memory
    ↪reallocation lv_table_set_set_value*/
    lv_table_set_col_cnt(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 %d", i + 1);
    }

    lv_obj_align(table, LV_ALIGN_CENTER, 0, -20);

    /*Add an event callback to apply some custom drawing*/
    lv_obj_add_event_cb(table, draw_event_cb, LV_EVENT_DRAW_PART_END, NULL);
    lv_obj_add_event_cb(table, change_event_cb, LV_EVENT_VALUE_CHANGED, NULL);

    lv_mem_monitor_t mon2;
    lv_mem_monitor(&mon2);

    uint32_t mem_used = mon1.free_size - mon2.free_size;

    uint32_t elaps = lv_tick_elaps(t);

    lv_obj_t * label = lv_label_create(lv_scr_act());
    lv_label_set_text_fmt(label, "%d items were created in %d ms\n"
                           "using %d bytes of memory",
                           ITEM_CNT, elaps, mem_used);

    lv_obj_align(label, LV_ALIGN_BOTTOM_MID, 0, -10);
}

```

(下页继续)

(续上页)

```
#endif
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/widgets/table/lv_example_table_2.py
```

MicroPython

No examples yet.

MicroPython

No examples yet.

API

Typedefs

```
typedef uint8_t lv_table_cell_ctrl_t
```

Enums

```
enum [anonymous]
```

Values:

```
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_EXPORT_CONST_INT(LV_TABLE_CELL_NONE)

*lv_obj_t *lv_table_create(lv_obj_t *parent)*

Create a table object

参数 parent -- pointer to an object, it will be the parent of the new table

返回 pointer to the created table

void lv_table_set_cell_value(*lv_obj_t* *obj, uint16_t row, uint16_t col, const char *txt)
Set the value of a cell.

注解: New rows/columns are added automatically if required

参数

- **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, uint16_t row, uint16_t col, const char *fmt, ...)
Set the value of a cell. Memory will be allocated to store the text by the table.

注解: New rows/columns are added automatically if required

参数

- **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_cnt(*lv_obj_t* *obj, uint16_t row_cnt)
Set the number of rows

参数

- **obj** -- table pointer to a Table object
- **row_cnt** -- number of rows

void lv_table_set_col_cnt(*lv_obj_t* *obj, uint16_t col_cnt)
Set the number of columns

参数

- **obj** -- table pointer to a Table object
- **col_cnt** -- number of columns.

void lv_table_set_col_width(*lv_obj_t* *obj, uint16_t col_id, lv_coord_t w)
Set the width of a column

参数

- **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_add_cell_ctrl(*lv_obj_t* *obj, uint16_t row, uint16_t col, *lv_table_cell_ctrl_t* ctrl)
Add control bits to the cell.

参数

- **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, uint16_t row, uint16_t col, *lv_table_cell_ctrl_t* ctrl)
Clear control bits of the cell.

参数

- **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

const char *lv_table_get_cell_value(*lv_obj_t* *obj, uint16_t row, uint16_t col)
Get the value of a cell.

参数

- **obj** -- pointer to a Table object
- **row** -- id of the row [0 .. row_cnt -1]
- **col** -- id of the column [0 .. col_cnt -1]

返回 text in the cell

uint16_t lv_table_get_row_cnt(*lv_obj_t* *obj)
Get the number of rows.

参数 **obj** -- table pointer to a Table object**返回** number of rows.

uint16_t lv_table_get_col_cnt(*lv_obj_t* *obj)
Get the number of columns.

参数 **obj** -- table pointer to a Table object**返回** number of columns.

lv_coord_t lv_table_get_col_width(*lv_obj_t* *obj, uint16_t col)
Get the width of a column

参数

- **obj** -- table pointer to a Table object
- **col** -- id of the column [0 .. LV_TABLE_COL_MAX -1]

返回 width of the column

bool lv_table_has_cell_ctrl(*lv_obj_t* *obj, uint16_t row, uint16_t col, *lv_table_cell_ctrl_t* ctrl)
Get whether a cell has the control bits

参数

- **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

返回 true: all control bits are set; false: not all control bits are set

```
void lv_table_get_selected_cell(lv_obj_t *obj, uint16_t *row, uint16_t *col)
Get the selected cell (pressed and or focused)
```

参数

- **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)

Variables

```
const lv_obj_class_t lv_table_class
struct lv_table_t
```

Public Members

```
lv_obj_t obj
uint16_t col_cnt
uint16_t row_cnt
char **cell_data
lv_coord_t *row_h
lv_coord_t *col_w
uint16_t col_act
uint16_t row_act
```

6.2.15 Text area (文本框) (lv_textarea)

Overview (概述)

The Text Area is a *Base object* with a *Label* and a cursor on it. Texts or characters can be added to it. Long lines are wrapped and when the text becomes long enough the Text area can be scrolled.

One line mode and password modes are supported.

文本框是一个*Base object*, 上面有一个*Label* 和一个光标。可以向其中添加文本或字符。长行被换行, 当文本变得足够长时, 可以滚动文本框。

支持单行模式和密码模式。

Parts and Styles (部件和样式)

- LV_PART_MAIN The background of the text area. Uses all the typical background style properties and the text related style properties including `text_align` to align the text to the left, right or center.
- LV_PART_SCROLLBAR The scrollbar that is shown when the text is too long.
- LV_PART_SELECTED Detemines the style of the `selected text`. Only `text_color` and `bg_color` style properties can be used. `bg_color` should be set directly on the label of the text area.
- LV_PART_CURSOR Marks the position where the characters are inserted. The cursor's area is always the bounding box of the current character. A block cursor can be created by adding a background color and background opacity to LV_PART_CURSOR's style. The create line cursor leave the cursor transparent and set a left border. The `anim_time` style property sets the cursor's blink time.
- LV_PART_TEXTAREA_PLACEHOLDER Unique to Text Area, allows styling the placeholder text.
- LV_PART_MAIN 文本框的背景。使用所有典型的背景样式属性和与文本相关的样式属性（包括“`text_align`”）将文本向左、向右或居中对齐。
- LV_PART_SCROLLBAR 文本过长时显示的滚动条。
- LV_PART_SELECTED 确定选定文本 的样式。只能使用 `text_color` 和 `bg_color` 样式属性。`bg_color` 应该直接设置在文本框的标签上。
- LV_PART_CURSOR 标记字符插入的位置。光标的区域始终是当前字符的边界框。可以通过向“LV_PART_CURSOR”的样式添加背景颜色和背景不透明度来创建块光标。创建行光标使光标保持透明并设置左边框。`anim_time` 样式属性设置光标的闪烁时间。
- LV_PART_TEXTAREA_PLACEHOLDER 文本框独有，允许设置占位符文本的样式。
-

Usage (用法)

Add text (添加文本)

You can insert text or characters to the current cursor's position with:

- `lv_textarea_add_char(textarea, 'c')`
- `lv_textarea_add_text(textarea, "insert this text")`

To add wide characters like 'á', 'ß' or CJK characters use `lv_textarea_add_text(ta, "á")`.

`lv_textarea_set_text(ta, "New text")` changes the whole text.

您可以使用以下命令在当前光标位置插入文本或字符：

- `lv_textarea_add_char(textarea, 'c')`
- `lv_textarea_add_text(textarea, " 插入此文本")`

要添加宽字符，如 'á'、'ß' 或 CJK 字符，请使用 `lv_textarea_add_text(ta, "á")`。

`lv_textarea_set_text(ta, "New text")` 改变整个文本。

Placeholder (占位符)

A placeholder text can be specified - which is displayed when the Text area is empty - with `lv_textarea_set_placeholder_text(ta, "Placeholder text")`

可以指定占位符文本 - 当文本框为空时显示 - 使用 `lv_textarea_set_placeholder_text(ta, "Placeholder text")`

Delete character (删除字符)

To delete a character from the left of the current cursor position use `lv_textarea_del_char(textarea)`. To delete from the right use `lv_textarea_del_char_forward(textarea)`

Move the cursor (移动光标)

The cursor position can be modified directly like `lv_textarea_set_cursor_pos(textarea, 10)`. The 0 position means "before the first characters", `LV_TA_CURSOR_LAST` means "after the last character"

You can step the cursor with

- lv_textarea_cursor_right(textarea)
 - lv_textarea_cursor_left(textarea)
 - lv_textarea_cursor_up(textarea)
 - lv_textarea_cursor_down(textarea)

If `lv_textarea_set_cursor_click_pos(textarea, true)` is applied the cursor will jump to the position where the Text area was clicked.

光标位置可以像 `lv_textarea_set_cursor_pos(textarea, 10)` 一样直接修改。0 位置表示“在第一个字符之前”，`LV_TEXTAREA_CURSOR_LAST` 表示“在最后一个字符之后”。

您可以使用

- `lv_textarea_cursor_right(textarea)`
 - `lv_textarea_cursor_left(textarea)`
 - `lv_textarea_cursor_up(textarea)`
 - `lv_textarea_cursor_down(textarea)`

如果应用 `lv_textarea_set_cursor_click_pos(textarea, true)`, 则光标将跳转到单击文本框的位置。

Hide the cursor (隐藏光标)

The cursor is always visible, however it can be a good idea to style it to be visible only in `LV_STATE_FOCUSED` state.
一般情况下光标始终可见，但最好将其样式设置为仅在“`LV_STATE_FOCUSED`”状态下可见。

One line mode (单行模式)

The Text area can be configured to be on a single line with `lv_textarea_set_one_line(textarea, true)`. In this mode the height is set automatically to show only one line, line break characters are ignored, and word wrap is disabled.

可以使用 `lv_textarea_set_one_line(textarea, true)` 将文本框配置为单行。在此模式下，高度自动设置为仅显示一行，忽略换行符，并禁用自动换行。

Password mode (密码模式)

The text area supports password mode which can be enabled with `lv_textarea_set_password_mode(textarea, true)`.

If the • (Bullet, U+2022) character exists in the font, the entered characters are converted to it after some time or when a new character is entered. If • not exists, * will be used.

In password mode `lv_textarea_get_text(textarea)` returns the actual text entered, not the bullet characters.

The visibility time can be adjusted with `LV_TEXTAREA_DEF_PWD_SHOW_TIME` in `lv_conf.h`.

文本框支持密码模式，可以通过 `lv_textarea_set_password_mode(textarea, true)` 启用。

如果字体中存在• (Bullet, U+2022) 字符，则将输入的字符转换为该字符一段时间后或输入新字符时。如果•不存在，*将被使用。

在密码模式下 `lv_textarea_get_text(textarea)` 返回输入的实际文本，而不是项目符号字符。

可见时间可以通过 `lv_conf.h` 中的 `LV_TEXTAREA_DEF_PWD_SHOW_TIME` 进行调整。

Accepted characters (字符白名单)

You can set a list of accepted characters with `lv_textareae_set_accepted_chars(textarea, "0123456789.+-")`. Other characters will be ignored.

您可以使用 `lv_textareae_set_accepted_chars(textarea, "0123456789.+-")` 设置接受字符列表。其他字符将被忽略。

Max text length (设置文本长度)

The maximum number of characters can be limited with `lv_textarea_set_max_length(textarea, max_char_num)`

可以使用 `lv_textarea_set_max_length(textarea, max_char_num)` 限制最大字符数

Very long texts (超长文本)

If there is a very long text in the Text area (e. g. > 20k characters), scrolling and drawing might be slow. However, by enabling `LV_LABEL_LONG_TXT_HINT 1` in `lv_conf.h` the performance can be hugely improved. This will save some additional information about the label to speed up its drawing. Using `LV_LABEL_LONG_TXT_HINT` the scrolling and drawing will be as fast as with "normal" short texts.

如果文本框中有很长的文本（例如> 20k 个字符），滚动和绘制可能会很慢。但是，通过在 `lv_conf.h` 中启用 `LV_LABEL_LONG_TXT_HINT 1`，性能可以得到极大的提高。这将保存有关标签的一些附加信息以加快其绘制速度。使用 `LV_LABEL_LONG_TXT_HINT` 滚动和绘图将与“普通”短文本一样快。

Select text (选择文本)

Any part of the text can be selected if enabled with `lv_textarea_set_text_selection(textarea, true)`. This works much like when you select text on your PC with your mouse.

如果启用了 `lv_textarea_set_text_selection(textarea, true)`，则可以选择文本的任何部分。这与您使用鼠标在 PC 上选择文本非常相似。

Events (事件)

- `LV_EVENT_INSERT` Sent right before a character or text is inserted. The event parameter is the text about to be inserted. `lv_textarea_set_insert_replace(textarea, "New text")` replaces the text to insert. The new text cannot be in a local variable which is destroyed when the event callback exists. "" means do not insert anything.
- `LV_EVENT_VALUE_CHANGED` Sent when the content of the text area has been changed.
- `LV_EVENT_APPLY` Sent when `LV_KEY_ENTER` is pressed (or sent) to a one line text area.

See the events of the [Base object](#) too.

Learn more about [Events](#).

- `LV_EVENT_INSERT` 在插入字符或文本之前发送。事件参数是即将插入的文本。`lv_textarea_set_insert_replace(textarea, "New text")` 替换要插入的文本。新文本不能位于当事件回调存在时被销毁的局部变量中。"" 表示不插入任何内容。
- `LV_EVENT_VALUE_CHANGED` 当文本框的内容改变时发送。
- `LV_EVENT_APPLY` 在按下 `LV_KEY_ENTER` 时发送（或（发送）到一行文本框。

参见[Base object](#) 的事件。

详细了解[事件](#)。

Keys (按键)

- `LV_KEY_UP/DOWN/LEFT/RIGHT` Move the cursor
- Any character Add the character to the current cursor position

Learn more about [Keys](#).

- `LV_KEY_UP/DOWN/LEFT/RIGHT` 移动光标
- 任意字符将字符添加到当前光标位置

了解有关[键](#)的更多信息。

Example

C

Simple Text area

```
#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(e);
    LV_LOG_USER("Enter was pressed. The current text is: %s", lv_textarea_get_text(ta));
}

static void btmn_event_handler(lv_event_t * e)
{
    lv_obj_t * obj = lv_event_get_target(e);
    lv_obj_t * ta = lv_event_get_user_data(e);
    const char * txt = lv_btmatrix_get_btn_text(obj, lv_btmatrix_get_selected_btn(obj));

    if(strcmp(txt, LV_SYMBOL_BACKSPACE) == 0) lv_textarea_del_char(ta);
    else if(strcmp(txt, LV_SYMBOL_NEW_LINE) == 0) lv_textarea_add_char(ta, '\n');
    else lv_textarea_add_text(ta, txt);
}

void lv_example_textarea_1(void)
{
    lv_obj_t * ta = lv_textarea_create(lv_scr_act());
    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 * btmn_map[] = {"1", "2", "3", "\n",
                                    "4", "5", "6", "\n",
                                    "7", "8", "9", "\n",
                                    LV_SYMBOL_BACKSPACE, "0", LV_SYMBOL_NEW_LINE, ""};

    lv_obj_t * btmn = lv_btmatrix_create(lv_scr_act());
    lv_obj_set_size(btmn, 200, 150);
    lv_obj_align(btmn, LV_ALIGN_BOTTOM_MID, 0, -10);
    lv_obj_add_event_cb(btmn, btmn_event_handler, LV_EVENT_VALUE_CHANGED, ta);
    lv_obj_clear_flag(btmn, LV_OBJ_FLAG_CLICK_FOCUSABLE); /*To keep the text area focused on button clicks*/
    lv_btmatrix_set_map(btmn, btmn_map);
}

#endif
```

```
def event_handler(obj, event):
    if event == lv.EVENT.VALUE_CHANGED:
        print("Value: %s" % obj.get_text())
```

(下页继续)

(续上页)

```

elif event == lv.EVENT.LONG_PRESSED_REPEAT:
    # For simple test: Long press the Text area to add the text below
    ta1.add_text("\n\nYou can scroll it if the text is long enough.\n")

ta1 = lv.ta(lv.scr_act())
ta1.set_size(200, 100)
ta1.align(None, lv.ALIGN.CENTER, 0, 0)
ta1.set_cursor_type(lv.CURSOR.BLOCK)
ta1.set_text("A text in a Text Area")      # Set an initial text
ta1.set_event_cb(event_handler)

```

Text area with password field

```

#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_scr_act());
    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_scr_act());
    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_scr_act());
    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_scr_act());
    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_scr_act());
    lv_obj_set_size(kb, LV_HOR_RES, LV_VER_RES / 2);
}

```

(下页继续)

(续上页)

```

    lv_keyboard_set_textarea(kb, pwd_ta); /*Focus it on one of the text areas to
→start*/
}

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(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) {
        const char * str = lv_event_get_param(e);
        if(str[0] == '\n') {
            LV_LOG_USER("Ready\n");
        }
    }
}

#endif

```

```

HOR_RES = lv.disp_get_hor_res(lv.disp_get_default())

def kb_event_cb(event_kb, event):
    # Just call the regular event handler
    event_kb.def_event_cb(event)

def ta_event_cb(ta, event):
    if event == lv.EVENT.INSERT:
        # get inserted value
        ptr = lv.C_Pointer()
        ptr.ptr_val = lv.event_get_data()
        if ptr.str_val == "\n":
            print("Ready")
    elif event == lv.EVENT.CLICKED:
        # Focus on the clicked text area
        kb.set_ta(ta)

# Create the password box
pwd_ta = lv.ta(lv.scr_act())
pwd_ta.set_text("");
pwd_ta.set_pwd_mode(True)
pwd_ta.set_one_line(True)
pwd_ta.set_width(HOR_RES // 2 - 20)
pwd_ta.set_pos(5, 20)
pwd_ta.set_event_cb(ta_event_cb)

# Create a label and position it above the text box
pwd_label = lv.label(lv.scr_act())
pwd_label.set_text("Password:")
pwd_label.align(pwd_ta, lv.ALIGN.OUT_TOP_LEFT, 0, 0)

# Create the one-line mode text area
oneline_ta = lv.ta(lv.scr_act(), pwd_ta)

```

(下页继续)

(续上页)

```

oneline_ta.set_pwd_mode(False)
oneline_ta.set_cursor_type(lv.CURSOR.LINE | lv.CURSOR.HIDDEN)
oneline_ta.align(None, lv.ALIGN.IN_TOP_RIGHT, -5, 20)
oneline_ta.set_event_cb(ta_event_cb)

# Create a label and position it above the text box
oneline_label = lv.label(lv.scr_act())
oneline_label.set_text("Text:")
oneline_label.align(oneline_ta, lv.ALIGN.OUT_TOP_LEFT, 0, 0)

# Create a keyboard and make it fill the width of the above text areas
kb = lv.kb(lv.scr_act())
kb.set_pos(5, 90)
kb.set_event_cb(kb_event_cb) # Setting a custom event handler stops the keyboard from
# closing automatically
kb.set_size(HOR_RES - 10, 140)

kb.set_ta(pwd_ta) # Focus it on one of the text areas to start
kb.set_cursor_manage(True) # Automatically show/hide cursors on text areas

```

Text auto-formatting

```

#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_scr_act());
    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_scr_act());
    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(e);
    const char * txt = lv_textarea_get_text(ta);
}

```

(下页继续)

(续上页)

```

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

```

Error encountered **while** trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/textarea/lv_example_textarea_3.py

MicroPython

No examples yet.

API

Enums

enum **[anonymous]**

Values:

enumerator **LV_PART_TEXTAREA_PLACEHOLDER**

Functions

LV_EXPORT_CONST_INT(LV_TEXTAREA_CURSOR_LAST)

*lv_obj_t *lv_textarea_create(lv_obj_t *parent)*

Create a text area objects

参数 parent -- pointer to an object, it will be the parent of the new text area

返回 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. 'Á' use _lv_txt_encoded_conv_wc('Á')

参数

- **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

参数

- **obj** -- pointer to a text area object

- **txt** -- a '\0' terminated string to insert

void lv_textarea_del_char(*lv_obj_t* *obj)

Delete a the left character from the current cursor position

参数 **obj** -- pointer to a text area object

void lv_textarea_del_char_forward(*lv_obj_t* *obj)

Delete the right character from the current cursor position

参数 **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

参数

- **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

参数

- **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

参数

- **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.

参数

- **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

参数

- **obj** -- pointer to a text area object
- **en** -- true: enable, false: disable

void lv_textarea_set_one_line(*lv_obj_t* *obj, bool en)

Configure the text area to one line or back to normal

参数

- **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

参数

- **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.

参数

- **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 an other text. It can be used to add automatic formatting to the text area.

参数

- **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.

参数

- **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, uint16_t time)

Set how long show the password before changing it to '*'.

参数

- **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)

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

参数

- **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).

参数 obj -- pointer to a text area object

返回 pointer to the text

const char *lv_textarea_get_placeholder_text(*lv_obj_t* *obj)

Get the placeholder text of a text area

参数 **obj** -- pointer to a text area object

返回 pointer to the text

`lv_obj_t *lv_textarea_get_label(const lv_obj_t *obj)`

Get the label of a text area

参数 **obj** -- pointer to a text area object

返回 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

参数 **obj** -- pointer to a text area object

返回 the cursor position

`bool lv_textarea_get_cursor_click_pos(lv_obj_t *obj)`

Get whether the cursor click positioning is enabled or not.

参数 **obj** -- pointer to a text area object

返回 true: enable click positions; false: disable

`bool lv_textarea_get_password_mode(const lv_obj_t *obj)`

Get the password mode attribute

参数 **obj** -- pointer to a text area object

返回 true: password mode is enabled, false: disabled

`bool lv_textarea_get_one_line(const lv_obj_t *obj)`

Get the one line configuration attribute

参数 **obj** -- pointer to a text area object

返回 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.

参数 **obj** -- pointer to a text area object

返回 list of accented characters.

`uint32_t lv_textarea_get_max_length(lv_obj_t *obj)`

Get max length of a Text Area.

参数 **obj** -- pointer to a text area object

返回 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.

参数 **obj** -- pointer to a text area object

返回 whether text is selected or not

`bool lv_textarea_get_text_selection(lv_obj_t *obj)`

Find whether selection mode is enabled.

参数 **obj** -- pointer to a text area object

返回 true: selection mode is enabled, false: disabled

`uint16_t lv_textarea_get_password_show_time(lv_obj_t *obj)`
Set how long show the password before changing it to '*'.

参数 **obj** -- pointer to a text area object

返回 show time in milliseconds. 0: hide immediately.

`void lv_textarea_clear_selection(lv_obj_t *obj)`
Clear the selection on the text area.

参数 **obj** -- pointer to a text area object

`void lv_textarea_cursor_right(lv_obj_t *obj)`
Move the cursor one character right

参数 **obj** -- pointer to a text area object

`void lv_textarea_cursor_left(lv_obj_t *obj)`
Move the cursor one character left

参数 **obj** -- pointer to a text area object

`void lv_textarea_cursor_down(lv_obj_t *obj)`
Move the cursor one line down

参数 **obj** -- pointer to a text area object

`void lv_textarea_cursor_up(lv_obj_t *obj)`
Move the cursor one line up

参数 **obj** -- pointer to a text area object

Variables

```
const lv_obj_class_t lv_textarea_class
struct lv_textarea_t
```

Public Members

```
lv_obj_t obj
lv_obj_t *label
char *placeholder_txt
char *pwd_tmp
const char *accepted_chars
uint32_t max_length
uint16_t pwd_show_time
lv_coord_t valid_x
uint32_t pos
lv_area_t area
uint32_t txt_byte_pos
uint8_t show
```

```

uint8_t click_pos
struct lv_textarea_t::[anonymous] cursor
uint32_t sel_start
uint32_t sel_end
uint8_t text_sel_in_prog
uint8_t text_sel_en
uint8_t pwd_mode
uint8_t one_line

```

6.3 Extra widgets (附加部件)

6.3.1 Calendar (日历) (lv_calendar)

Overview (概述)

The Calendar object is a classic calendar which can:

- show the days of any month in a 7x7 matrix
- Show the name of the days
- highlight the current day (today)
- highlight any user-defined dates

The Calendar is added to the default group (if it is set). Calendar is an editable object which allow selecting and clicking the dates with encoder navigation too.

To make the Calendar flexible, by default it doesn't show the current year or month. Instead, there are external "headers" that can be attached to the calendar.

Calendar 对象是一个经典的日历，它可以：

- 在 7x7 矩阵中显示任何月份的天数
- 显示日期的名称
- 突出显示当天（今天）
- 突出显示任何用户定义的日期

日历将添加到默认组（如果已设置）。日历是一个可编辑的对象，它也允许使用编码器导航选择和单击日期。

为了使日历灵活，默认情况下它不显示当前的年份或月份。相反，有可以附加到日历的外部“标题”。

Parts and Styles (部件和样式)

The calendar object uses the `Button matrix` object under the hood to arrange the days into a matrix.

- `LV_PART_MAIN` The background of the calendar. Uses all the background related style properties.
- `LV_PART_ITEMS` Refers to the dates and day names. Button matrix control flags are set to differentiate the buttons and a custom drawer event is added modify the properties of the buttons as follows:
 - day names have no border, no background and drawn with a gray color
 - days of the previous and next month have `LV_BTNMATRIX_CTRL_DISABLED` flag
 - today has a thicker border with the theme's primary color
 - highlighted days have some opacity with the theme's primary color.

日历对象使用引擎盖下的`Button matrix`对象将日期排列成矩阵。

- `LV_PART_MAIN` 日历的背景。使用所有与背景相关的样式属性。
- `LV_PART_ITEMS` 指的是日期和日期名称。设置按钮矩阵控制标志以区分按钮并添加自定义抽屉事件，修改按钮的属性如下：
 - 日名称没有边框，没有背景，并用灰色绘制
 - 上个月和下个月的天数有 `LV_BTNMATRIX_CTRL_DISABLED` 标志
 - 今天与主题的主色有更厚的边框
 - 突出显示的日子与主题的主要颜色有一些不透明度。

Usage (用法)

Some functions use the `lv_calendar_date_t` type which is a structure with `year`, `month` and `day` fields.

一些函数使用 `lv_calendar_date_t` 类型，这是一个带有 `year`、`month` 和 `day` 字段的结构。

Current date (当前日期)

To set the current date (today), use the `lv_calendar_set_today_date(calendar, year, month, day)` function. `month` needs to be in 1..12 range and `day` in 1..31 range.

要设置当前日期（今天），请使用 `lv_calendar_set_today_date(calendar, year, month, day)` 函数。`month` 需要在 1..12 范围内，`day` 需要在 1..31 范围内。

Shown date (显示日期)

To set the shown date, use `lv_calendar_set_shown_date(calendar, year, month);`

要设置显示日期，请使用 `lv_calendar_set_shown_date(calendar, year, month);`

Highlighted days (突出显示的日子)

The list of highlighted dates should be stored in a `lv_calendar_date_t` array loaded by `lv_calendar_set_highlighted_dates(calendar, highlighted_dates, date_num)`. Only the array's pointer will be saved so the array should be a static or global variable.

突出显示的日期列表应存储在由 `lv_calendar_set_highlighted_dates(calendar, highlight_dates, date_num)` 加载的 `lv_calendar_date_t` 数组中。只有数组的指针会被保存，所以数组应该是一个静态或全局变量。

Name of the days (日期)

The name of the days can be adjusted with `lv_calendar_set_day_names(calendar, day_names)` where `day_names` looks like `const char * day_names[7] = {"Su", "Mo", ...};` Only the pointer of the day names is saved so the elements should be static, global or constant variables.

日期的名称可以用 `lv_calendar_set_day_names(calendar, day_names)` 调整，其中 `day_names` 看起来像 `const char * day_names[7] = {"Su", "Mo", ...};` 只保存日期名称的指针，因此元素应该是静态、全局或常量变量。

Events (事件)

- `LV_EVENT_VALUE_CHANGED` Sent if a date is clicked. `lv_calendar_get_pressed_date(calendar, &date)` set `date` to the date currently being pressed. Returns `LV_RES_OK` if there is a valid pressed date, else `LV_RES_INV`.

Learn more about [Events](#).

- `LV_EVENT_VALUE_CHANGED` 如果点击日期发送。`lv_calendar_get_pressed_date(calendar, &date)` 将 `date` 设置为当前按下的日期。如果存在有效的按下日期，则返回 `LV_RES_OK`，否则返回 `LV_RES_INV`。

详细了解[事件](#)。

Keys (按键)

- `LV_KEY_RIGHT/UP/LEFT/RIGHT` To navigate among the buttons to dates
- `LV_KEY_ENTER` To press/release the selected date

Learn more about [Keys](#).

- `LV_KEY_RIGHT/UP/LEFT/RIGHT` 在按钮之间导航到日期
- `LV_KEY_ENTER` 按下/释放所选日期

了解有关[Keys](#) 的更多信息。

Headers (标题)

Arrow buttons (箭头按钮)

`lv_calendar_header_arrow_create(parent, calendar, button_size)` creates a header that contains a left and right arrow on the sides and a text with the current year and month between them.

`lv_calendar_header_arrow_create(parent, calendar, button_size)` 创建一个标题，其中包含两侧的左右箭头和文本，其中包含当前年份和月份。

Drop-down (下拉)

`lv_calendar_header_dropdown_create(parent, calendar)` creates a header that contains 2 drop-down lists: one for the year and another for the month.

`lv_calendar_header_dropdown_create(parent, calendar)` 创建一个包含 2 个下拉列表的标题：一个用于年份，另一个用于月份。

Example

C

Calendar with header

```
#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_event_get_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_scr_act());
    lv_obj_set_size(calendar, 200, 200);
    lv_obj_align(calendar, LV_ALIGN_CENTER, 0, 20);
    lv_obj_add_event_cb(calendar, event_handler, LV_EVENT_ALL, NULL);

    lv_calendar_set_today_date(calendar, 2021, 02, 23);
    lv_calendar_set_showed_date(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*/
}
```

(下页继续)

(续上页)

```

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_header_dropdown_create(lv_scr_act(), calendar);
#elif LV_USE_CALENDAR_HEADER_ARROW
    lv_calendar_header_arrow_create(lv_scr_act(), calendar, 25);
#endif
}

#endif

```

Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/widgets/calendar/lv_example_calendar_1.py

MicroPython

No examples yet.

API

Functions

`lv_obj_t *lv_calendar_create(lv_obj_t *parent)`

`void lv_calendar_set_today_date(lv_obj_t *obj, uint32_t year, uint32_t month, uint32_t day)`

Set the today's date

参数

- **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_showed_date(lv_obj_t *obj, uint32_t year, uint32_t month)`

Set the currently showed

参数

- **obj** -- pointer to a calendar object

- **year** -- today's year
- **month** -- today's month [1..12]

```
void lv_calendar_set_highlighted_dates(lv_obj_t *obj, lv_calendar_date_t highlighted[], uint16_t date_num)
```

Set the the highlighted dates

参数

- **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

参数

- **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.

```
const lv_calendar_date_t *lv_calendar_get_today_date(const lv_obj_t *calendar)
```

Get the today's date

参数 **calendar** -- pointer to a calendar object

返回 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

参数 **calendar** -- pointer to a calendar object

返回 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 the highlighted dates

参数 **calendar** -- pointer to a calendar object

返回 pointer to an `lv_calendar_date_t` array containing the dates.

```
uint16_t lv_calendar_get_highlighted_dates_num(const lv_obj_t *calendar)
```

Get the number of the highlighted dates

参数 **calendar** -- pointer to a calendar object

返回 number of highlighted days

```
lv_res_t lv_calendar_get_pressed_date(const lv_obj_t *calendar, lv_calendar_date_t *date)
```

Get the currently pressed day

参数

- **calendar** -- pointer to a calendar object
- **date** -- store the pressed date here

返回 LV_RES_OK: there is a valid pressed date; LV_RES_INV: 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
int8_t month
int8_t day
    1..12

struct lv_calendar_t
```

Public Members

```
lv_btnmatrix_t btm
lv_calendar_date_t today
lv_calendar_date_t showed_date
lv_calendar_date_t *highlighted_dates
uint16_t highlighted_dates_num
const char *map[8 * 7]
char nums[7 * 6][4]
```

6.3.2 Chart (lv_chart)

Overview

Charts are a basic object to visualize data points. Currently *Line* charts (connect points with lines and/or draw points on them) and *Bar* charts are supported.

Charts can have:

- division lines
- 2 y axis
- axis ticks and texts on ticks
- cursors
- scrolling and zooming

Parts and Styles

- **LV_PART_MAIN** The background of the chart. Uses all the typical background and *line* (for the division lines) related style properties. *Padding* makes the series area smaller.
- **LV_PART_SCROLLBAR** The scrollbar used if the chart is zoomed. See the *Base object*'s documentation for details.
- **LV_PART_ITEMS** Refers to the line or bar series.
 - Line chart: The *line* properties are used by the lines. **width**, **height**, **bg_color** and **radius** is used to set the appearance of points.
 - Bar chart: The typical background properties are used to style the bars.
- **LV_PART_INDICATOR** Refers to the points on line and scatter chart (small circles or squares).
- **LV_PART_CURSOR** *Line* properties are used to style the cursors. **width**, **height**, **bg_color** and **radius** are used to set the appearance of points.
- **LV_PART_TICKS** *Line* and *Text* style properties are used to style the ticks

Usage

Chart type

The following data display types exist:

- **LV_CHART_TYPE_NONE** Do not display any data. Can be used to hide the series.
- **LV_CHART_TYPE_LINE** Draw lines between the data points and/or points (rectangles or circles) on the data points.
- **LV_CHART_TYPE_BAR** - Draw bars.
- **LV_CHART_TYPE_SCATTER** - X/Y chart drawing point's and lines between the points. .

You can specify the display type with `lv_chart_set_type(chart, LV_CHART_TYPE_...)`.

Data series

You can add any number of series to the charts by `lv_chart_add_series(chart, color, axis)`. This will allocates a `lv_chart_series_t` structure which contains the chosen `color` and an array for the data points. `axis` can have the following values:

- **LV_CHART_AXIS_PRIMARY_Y** Left axis
- **LV_CHART_AXIS_SECONDARY_Y** Right axis
- **LV_CHART_AXIS_PRIMARY_X** Bottom axis
- **LV_CHART_AXIS_SECONDARY_X** Top axis

`axis` tells which axis's range should be used te scale the values.

`lv_chart_set_ext_y_array(chart, ser, value_array)` makes the chart use an external array for the given series. `value_array` should look like this: `lv_coord_t * value_array[num_points]`. The array size needs to be large enough to hold all the points of that series. The array's pointer will be saved in the chart so it needs to be global, static or dynamically allocated. Note: you should call `lv_chart_refresh(chart)` after the external data source has been updated to update the chart.

The value array of a series can be obtained with `lv_chart_get_y_array(chart, ser)`, which can be used with `ext_array` or *normal arrays*.

For `LV_CHART_TYPE_SCATTER` type `lv_chart_set_ext_x_array(chart, ser, value_array)` and `lv_chart_get_x_array(chart, ser)` can be used as well.

Modify the data

You have several options to set the data of series:

1. Set the values manually in the array like `ser1->points[3] = 7` and refresh the chart with `lv_chart_refresh(chart)`.
2. Use `lv_chart_set_value_by_id(chart, ser, value, id)` where `id` is the index of the point you wish to update.
3. Use the `lv_chart_set_next_value(chart, ser, value)`.
4. Initialize all points to a given value with: `lv_chart_set_all_value(chart, ser, value)`.

Use `LV_CHART_POINT_DEF` as value to make the library skip drawing that point, column, or line segment.

For `LV_CHART_TYPE_SCATTER` type `lv_chart_set_value_by_id2(chart, ser, id, value)` and `lv_chart_set_next_value2(chart, ser, x_valuem y_value)` can be used as well.

Update modes

`lv_chart_set_next_value` can behave in two ways depending on *update mode*:

- `LV_CHART_UPDATE_MODE_SHIFT` Shift old data to the left and add the new one to the right.
- `LV_CHART_UPDATE_MODE_CIRCULAR` - Add the new data in circular fashion, like an ECG diagram).

The update mode can be changed with `lv_chart_set_update_mode(chart, LV_CHART_UPDATE_MODE_...)`.

Number of points

The number of points in the series can be modified by `lv_chart_set_point_count(chart, point_num)`. The default value is 10. Note: this also affects the number of points processed when an external buffer is assigned to a series, so you need to be sure the external array is large enough.

Handling large number of points

On line charts if the number of points is greater than the pixels horizontally, the Chart will draw only vertical lines to make the drawing of large amount of data effective. If there are, let's say, 10 points to a pixel, LVGL searches the smallest and the largest value and draws a vertical lines between them to ensure no peaks are missed.

Vertical range

You can specify the minimum and maximum values in y-direction with `lv_chart_set_range(chart, axis, min, max)`. `axis` can be `LV_CHART_AXIS_PRIMARY` (left axis) or `LV_CHART_AXIS_SECONDARY` (right axis).

The value of the points will be scaled proportionally. The default range is: 0..100.

Division lines

The number of horizontal and vertical division lines can be modified by `lv_chart_set_div_line_count(chart, hdiv_num, vdiv_num)`. The default settings are 3 horizontal and 5 vertical division lines. If there is a visible border on a side and no padding on that side, the division line would be drawn on top of the border and therefore it won't be drawn.

Override default start point for series

If you want a plot to start from a point other than the default which is `point[0]` of the series, you can set an alternative index with the function `lv_chart_set_x_start_point(chart, ser, id)` where `id` is the new index position to start plotting from.

Note that `LV_CHART_UPDATE_MODE_SHIFT` also changes the `start_point`.

Tick marks and labels

Ticks and labels can be added to the axis with `lv_chart_set_axis_tick(chart, axis, major_len, minor_len, major_cnt, minor_cnt, label_en, draw_size)`.

- `axis` can be `LV_CHART_AXIS_X/PRIMARY_Y/SECONDARY_Y`
- `major_len` is the length of major ticks
- `minor_len` is the length of minor ticks
- `major_cnt` is the number of major ticks on the axis
- `minor_cnt` is the number of minor ticks between two major ticks
- `label_en true`: enable label drawing on major ticks
- `draw_size` extra size required to draw the tick and labels (start with 20 px and increase if the ticks/labels are clipped)

Zoom

The chart can be zoomed independently in x and y directions with `lv_chart_set_zoom_x(chart, factor)` and `lv_chart_set_zoom_y(chart, factor)`. If `factor` is 256 there is no zoom. 512 means double zoom, etc. Fractional values are also possible but < 256 value is not allowed.

Cursor

A cursor can be added with `lv_chart_cursor_t * c1 = lv_chart_add_cursor(chart, color, dir);`. The possible values of `dir` `LV_DIR_NONE/RIGHT/UP/LEFT/DOWN/HOR/VER/ALL` or their OR-ed values to tell in which direction(s) should the cursor be drawn.

`lv_chart_set_cursor_pos(chart, cursor, &point)` sets the position of the cursor. `pos` is a pointer to an `lv_point_t` variable. E.g. `lv_point_t point = {10, 20};`. If the chart is scrolled the cursor will remain in the same place.

`lv_chart_get_point_pos_by_id(chart, series, id, &point_out)` gets the coordinate of a given point. It's useful to place the cursor at a given point.

`lv_chart_set_cursor_point(chart, cursor, series, point_id)` sticks the cursor at a point. If the point's position changes (new value or scrolling) the cursor will move with the point.

Events

- `LV_EVENT_VALUE_CHANGED` Sent when a new point is clicked or pressed. `lv_chart_get_pressed_point(chart)` returns the zero-based index of the pressed point.
- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` are sent with the following types:
 - `LV_CHART_DRAW_PART_DIV_LINE_INIT` Used before/after drawn the div lines to add masks to any extra drawings. The following fields are set:
 - * `part: LV_PART_MAIN`
 - * `line_dsc`
 - `LV_CHART_DRAW_PART_DIV_LINE_HOR`, `LV_CHART_DRAW_PART_DIV_LINE_VER` Used for each horizontal and vertical division lines.
 - * `part: LV_PART_MAIN`
 - * `id: index of the line`
 - * `p1, p2: points of the line`
 - * `line_dsc`
 - `LV_CHART_DRAW_PART_LINE_AND_POINT` Used on line and scatter charts for lines and points.
 - * `part: LV_PART_ITEMS`
 - * `id: index of the point`
 - * `value: value of idth point`
 - * `p1, p2: points of the line`
 - * `draw_area: area of the point`
 - * `line_dsc`
 - * `rect_dsc`
 - * `sub_part_ptr: pointer to the series`
 - `LV_CHART_DRAW_PART_BAR` Used on bar charts for the rectangles.
 - * `part: LV_PART_ITEMS`
 - * `id: index of the point`

- * `value`: value of `id`th point
- * `draw_area`: area of the point
- * `rect_dsc`:
- * `sub_part_ptr`: pointer to the series
- `LV_CHART_DRAW_PART_CURSOR` Used on cursor lines and points.
 - * `part`: `LV_PART_CURSOR`
 - * `p1, p2`: points of the line
 - * `line_dsc`
 - * `rect_dsc`
 - * `draw_area`: area of the points
- `LV_CHART_DRAW_PART_TICK_LABEL` Used on tick lines and labels.
 - * `part`: `LV_PART_TICKS`
 - * `id`: axis
 - * `value`: value of the tick
 - * `text`: `value` converted to decimal or `NULL` for minor ticks
 - * `line_dsc`,
 - * `label_dsc`,

See the events of the [Base object](#) too.

Learn more about [Events](#).

Keys

No *Keys* are processed by the object type.

Learn more about [Keys](#).

Example

C

Line Chart

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↳ lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_chart/lv_ex_chart_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↳ lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_chart/lv_ex_chart_1.py
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↳ lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_chart/lv_ex_chart_2.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_chart/lv_ex_chart_2.py
```

MicroPython

No examples yet.

API

Typedefs

```
typedef uint8_t lv_chart_type_t
typedef uint8_t lv_chart_update_mode_t
typedef uint8_t lv_chart_axis_t
```

Enums

enum [anonymous]

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 columns

enum [anonymous]

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 [anonymous]

Enumeration of the axis'

Values:

enumerator LV_CHART_AXIS_PRIMARY_Y

enumerator **LV_CHART_AXIS_SECONDARY_Y**
 enumerator **LV_CHART_AXIS_X**
 enumerator **_LV_CHART_AXIS_LAST**

Functions

LV_EXPORT_CONST_INT(LV_CHART_POINT_NONE)

*lv_obj_t *lv_chart_create(lv_obj_t *parent)*

Create a chart objects

参数 **parent** -- pointer to an object, it will be the parent of the new button

返回 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

参数

- **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, uint16_t cnt)

Set the number of points on a data line on a chart

参数

- **obj** -- pointer r to chart object
- **cnt** -- new number of points on the data lines

void lv_chart_set_range(lv_obj_t *obj, lv_chart_axis_t axis, lv_coord_t min, lv_coord_t max)

Set the minimal and maximal y values on an axis

参数

- **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_update_mode(lv_obj_t *obj, lv_chart_update_mode_t update_mode)

Set update mode of the chart object. Affects

参数

- **obj** -- pointer to a chart object
- **mode** -- the update mode

void lv_chart_set_div_line_count(lv_obj_t *obj, uint8_t hdiv, uint8_t vdiv)

Set the number of horizontal and vertical division lines

参数

- **obj** -- pointer to a chart object
- **hdiv** -- number of horizontal division lines

- **vdiv** -- number of vertical division lines

void lv_chart_set_zoom_x(*lv_obj_t* *obj, uint16_t zoom_x)
Zoom into the chart in X direction

参数

- **obj** -- pointer to a chart object
- **zoom_x** -- zoom in x direction. LV_ZOOM_NONE or 256 for no zoom, 512 double zoom

void lv_chart_set_zoom_y(*lv_obj_t* *obj, uint16_t zoom_y)
Zoom into the chart in Y direction

参数

- **obj** -- pointer to a chart object
- **zoom_y** -- zoom in y direction. LV_ZOOM_NONE or 256 for no zoom, 512 double zoom

uint16_t lv_chart_get_zoom_x(const *lv_obj_t* *obj)
Get X zoom of a chart

参数 **obj** -- pointer to a chart object

返回 the X zoom value

uint16_t lv_chart_get_zoom_y(const *lv_obj_t* *obj)
Get Y zoom of a chart

参数 **obj** -- pointer to a chart object

返回 the Y zoom value

void lv_chart_set_axis_tick(*lv_obj_t* *obj, *lv_chart_axis_t* axis, lv_coord_t major_len, lv_coord_t minor_len, lv_coord_t major_cnt, lv_coord_t minor_cnt, bool label_en, lv_coord_t draw_size)

Set the number of tick lines on an axis

参数

- **obj** -- pointer to a chart object
- **axis** -- an axis which ticks count should be set
- **major_len** -- length of major ticks
- **minor_len** -- length of minor ticks
- **major_cnt** -- number of major ticks on the axis
- **minor_cnt** -- number of minor ticks between two major ticks
- **label_en** -- true: enable label drawing on major ticks
- **draw_size** -- extra size required to draw the tick and labels (start with 20 px and increase if the ticks/labels are clipped)

lv_chart_type_t **lv_chart_get_type(const *lv_obj_t* *obj)**
Get the type of a chart

参数 **obj** -- pointer to chart object

返回 type of the chart (from '*lv_chart_t*' enum)

uint16_t lv_chart_get_point_count(const *lv_obj_t* *obj)
Get the data point number per data line on chart

参数 **chart** -- pointer to chart object

返回 point number on each data line

```
uint16_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
```

参数

- **chart** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'

返回 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, uint16_t id, lv_point_t *p_out)
Get the position of point of the an index relative to the chart.
```

参数

- **chart** -- 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

参数 **chart** -- 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

参数

- **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)

返回 pointer to the allocated data series

```
void lv_chart_remove_series(lv_obj_t *obj, lv_chart_series_t *series)
```

Deallocate and remove a data series from a chart

参数

- **chart** -- 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.

参数

- **obj** -- 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

参数

- **obj** -- pointer to a chart object.
- **series** -- pointer to a series object
- **color** -- the new color of the series

`void lv_chart_set_x_start_point(lv_obj_t *obj, lv_chart_series_t *ser, uint16_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.

参数

- **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.

参数

- **chart** -- pointer to a chart
- **ser** -- the previous series or NULL to get the first

返回 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

参数

- **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

返回 pointer to the created cursor

`void lv_chart_set_cursor_point(lv_obj_t *chart, lv_chart_cursor_t *cursor, lv_point_t *point)`

Set the coordinate of the cursor with respect to the paddings

参数

- **obj** -- pointer to a chart object
- **cursor** -- pointer to the cursor
- **point** -- the new coordinate of cursor relative to paddings of the background

`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

参数

- **obj** -- pointer to a chart object
- **cursor** -- pointer to cursor

返回 coordinate of the cursor as lv_point_t

`void lv_chart_set_all_value(lv_obj_t *obj, lv_chart_series_t *ser, lv_coord_t value)`

Initialize all data points of a series with a value

参数

- **obj** -- pointer to chart object
- **ser** -- pointer to a data series on 'chart'
- **value** -- the new value for all points. LV_CHART_POINT_DEF can be used to hide the points.

void lv_chart_set_next_value(*lv_obj_t* *obj, *lv_chart_series_t* *ser, *lv_coord_t* value)

Set the next point according to the update mode policy.

参数

- **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_value_by_id(*lv_obj_t* *obj, *lv_chart_series_t* *ser, *lv_coord_t* value, *uint16_t* id)

Set an individual point's y value of a chart's series directly based on its index

参数

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **value** -- value to assign to array point
- **id** -- the index of the x point in the array

void lv_chart_set_ext_array(*lv_obj_t* *obj, *lv_chart_series_t* *ser, *lv_coord_t* array[])

Set an external array of data points to use for the chart NOTE: It is the users responsibility to make sure the **point_cnt** matches the external array size.

参数

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'
- **array** -- external array of points for chart

***lv_coord_t* *lv_chart_get_array(const *lv_obj_t* *obj, *lv_chart_series_t* *ser)**

Get the array of values of a series

参数

- **obj** -- pointer to a chart object
- **ser** -- pointer to a data series on 'chart'

返回 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.

参数 **obj** -- pointer to a chart object

返回 the index of the point [0 .. point count] or LV_CHART_POINT_ID_NONE if no point is being pressed

Variables

```
const lv_obj_class_t lv_chart_class
struct lv_chart_series_t
    #include <lv_chart.h> Descriptor a chart series
```

Public Members

```
lv_coord_t *points
lv_color_t color
uint16_t last_point
uint8_t hidden
uint8_t ext_buf_assigned
lv_chart_axis_t y_axis
struct lv_chart_cursor_t
```

Public Members

```
lv_point_t point
lv_color_t color
lv_dir_t dir
struct lv_chart_tick_dsc_t
```

Public Members

```
lv_coord_t major_len
lv_coord_t minor_len
lv_coord_t draw_size
uint32_t minor_cnt
uint32_t major_cnt
uint32_t label_en
struct lv_chart_t
```

Public Members

lv_obj_t **obj**

lv_ll_t **series_ll**

Linked list for the series (stores *lv_chart_series_t*)

lv_ll_t **cursor_ll**

Linked list for the cursors (stores *lv_chart_cursor_t*)

lv_chart_tick_dsc_t **tick[LV_CHART_AXIS_LAST]**

lv_coord_t **ymin[2]**

lv_coord_t **ymax[2]**

uint16_t **pressed_point_id**

uint16_t **hdiv_cnt**

Number of horizontal division lines

uint16_t **vdiv_cnt**

Number of vertical division lines

uint16_t **point_cnt**

Point number in a data line

uint16_t **zoom_x**

uint16_t **zoom_y**

lv_chart_type_t **type**

Line or column chart

lv_chart_update_mode_t **update_mode**

6.3.3 Color wheel (*lv_colorwheel*)

Overview

As its name implies *Color wheel* allows the user to select a color. The Hue, Saturation and Value of the color can be selected separately.

Long pressing the object, the color wheel will change to the next parameter of the color (hue, saturation or value). A double click will reset the current parameter.

Parts and Styles

- LV_PART_MAIN Only `arc_width` is used to set the width of the color wheel
- LV_PART_KNOB A rectangle (or circle) drawn on the current value. It uses all the rectangle like style properties and padding to make it larger than the width of the arc.

Usage

Create a color wheel

`lv_colorwheel_create(parent, knob_recolor)` creates a new color wheel. With `knob_recolor=true` the knob's background color will be set to the current color.

Set color

The color can be set manually with `lv_colorwheel_set_hue/saturation/value(colorwheel, x)` or all at once with `lv_colorwheel_set_hsv(colorwheel, hsv)` or `lv_colorwheel_set_color(colorwheel, rgb)`

Color mode

The current color mode can be manually selected with `lv_colorwheel_set_color_mode(colorwheel, LV_COLORWHEEL_MODE_HUE/SATURATION/VALUE)`.

The color mode can be fixed (so as to not change with long press) using `lv_colorwheel_set_color_mode_fixed(colorwheel, true)`

Events

- LV_EVENT_VALUE_CHANGED Sent if a new color is selected.

Learn more about [Events](#).

Keys

- LV_KEY_UP, LV_KEY_RIGHT Increment the current parameter's value by 1
- LV_KEY_DOWN, LV_KEY_LEFT Decrement the current parameter's by 1
- LV_KEY_ENTER A long press will show the next mode. Double click to reset the current parameter.

Learn more about [Keys](#).

Example

API

Typedefs

```
typedef uint8_t lv_colorwheel_mode_t
```

Enums

```
enum [anonymous]
```

Values:

- enumerator **LV_COLORWHEEL_MODE_HUE**
- enumerator **LV_COLORWHEEL_MODE_SATURATION**
- enumerator **LV_COLORWHEEL_MODE_VALUE**

Functions

`lv_obj_t *lv_colorwheel_create(lv_obj_t *parent, bool knob_recolor)`

Create a color picker objects with disc shape

参数

- **parent** -- pointer to an object, it will be the parent of the new color picker
- **knob_recolor** -- true: set the knob's color to the current color

返回 pointer to the created color picker

`bool lv_colorwheel_set_hsv(lv_obj_t *obj, lv_color_hsv_t hsv)`

Set the current hsv of a color wheel.

参数

- **colorwheel** -- pointer to color wheel object
- **color** -- current selected hsv

返回 true if changed, otherwise false

`bool lv_colorwheel_set_rgb(lv_obj_t *obj, lv_color_t color)`

Set the current color of a color wheel.

参数

- **colorwheel** -- pointer to color wheel object
- **color** -- current selected color

返回 true if changed, otherwise false

`void lv_colorwheel_set_mode(lv_obj_t *obj, lv_colorwheel_mode_t mode)`

Set the current color mode.

参数

- **colorwheel** -- pointer to color wheel object

- **mode** -- color mode (hue/sat/val)

void lv_colorwheel_set_mode_fixed(lv_obj_t *obj, bool fixed)
Set if the color mode is changed on long press on center

参数

- **colorwheel** -- pointer to color wheel object
- **fixed** -- color mode cannot be changed on long press

lv_color_hsv_t lv_colorwheel_get_hsv(lv_obj_t *obj)
Get the current selected hsv of a color wheel.

参数 **colorwheel** -- pointer to color wheel object

返回 current selected hsv

lv_color_t lv_colorwheel_get_rgb(lv_obj_t *obj)
Get the current selected color of a color wheel.

参数 **colorwheel** -- pointer to color wheel object

返回 color current selected color

lv_colorwheel_mode_t lv_colorwheel_get_color_mode(lv_obj_t *obj)
Get the current color mode.

参数 **colorwheel** -- pointer to color wheel object

返回 color mode (hue/sat/val)

bool lv_colorwheel_get_color_mode_fixed(lv_obj_t *obj)
Get if the color mode is changed on long press on center

参数 **colorwheel** -- pointer to color wheel object

返回 mode cannot be changed on long press

Variables

```
const lv_obj_class_t lv_colorwheel_class
struct lv_colorwheel_t
```

Public Members

```
lv_obj_t obj
lv_color_hsv_t hsv
lv_point_t pos
uint8_t recolor
struct lv_colorwheel_t::[anonymous] knob
uint32_t last_click_time
uint32_t last_change_time
lv_point_t last_press_point
```

`lv_colorwheel_mode_t mode`

`uint8_t mode_fixed`

6.3.4 Image button (lv_imgbtn)

Overview

The Image button is very similar to the simple 'Button' object. The only difference is that it displays user-defined images in each state instead of drawing a rectangle.

You can set a left, right and center image, and the center image will be repeated to match the width of the object.

Parts and Styles

- `LV_PART_MAIN` Refers to the image(s). If background style properties are used, a rectangle will be drawn behind the image button.

Usage

Image sources

To set the image in a state, use the `lv_imgbtn_set_src(imgbtn, LV_IMGBTN_STATE_..., src_left, src_center, src_right)`.

The image sources work the same as described in the [Image object](#) except that "Symbols" are not supported by the Image button. Any of the sources can `NULL`.

The possible states are:

- `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`

If you set sources only in `LV_IMGBTN_STATE_RELEASED`, these sources will be used in other states too. If you set e.g. `LV_IMGBTN_STATE_PRESSED` they will be used in pressed state instead of the released images.

Events

- `LV_EVENT_VALUE_CHANGED` Sent when the button is toggled.

Learn more about [Events](#).

Keys

- LV_KEY_RIGHT/UP Go to toggled state if LV_OBJ_FLAG_CHECKABLE is enabled.
- LV_KEY_LEFT/DOWN Go to non-toggled state if LV_OBJ_FLAG_CHECKABLE is enabled.
- LV_KEY_ENTER Clicks the button

Learn more about [Keys](#).

Example

C

Simple Image button

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_imgbtn/lv_ex_imgbtn_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_imgbtn/lv_ex_imgbtn_1.py
```

MicroPython

No examples yet.

API

Enums

enum **lv_imbtn_state_t**

Values:

```
enumerator LV_IMGBTN_STATE_RELEASED
enumerator LV_IMGBTN_STATE_PRESSED
enumerator LV_IMGBTN_STATE_DISABLED
enumerator LV_IMGBTN_STATE_CHECKED_RELEASED
enumerator LV_IMGBTN_STATE_CHECKED_PRESSED
enumerator LV_IMGBTN_STATE_CHECKED_DISABLED
enumerator _LV_IMGBTN_STATE_NUM
```

Functions

`lv_obj_t *lv_imgbtn_create(lv_obj_t *parent)`

Create a image button objects

参数 **par** -- pointer to an object, it will be the parent of the new image button

返回 pointer to the created image button

`void lv_imgbtn_set_src(lv_obj_t *imgbtn, lv_imgbtn_state_t state, const void *src_left, const void *src_mid, const void *src_right)`

Set images for a state of the image button

参数

- **imgbtn** -- 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)

`const void *lv_imgbtn_get_src_left(lv_obj_t *imgbtn, lv_imgbtn_state_t state)`

Get the left image in a given state

参数

- **imgbtn** -- pointer to an image button object
- **state** -- the state where to get the image (from `lv_btn_state_t`)

返回 pointer to the left image source (a C array or path to a file)

`const void *lv_imgbtn_get_src_middle(lv_obj_t *imgbtn, lv_imgbtn_state_t state)`

Get the middle image in a given state

参数

- **imgbtn** -- pointer to an image button object
- **state** -- the state where to get the image (from `lv_btn_state_t`)

返回 pointer to the middle image source (a C array or path to a file)

`const void *lv_imgbtn_get_src_right(lv_obj_t *imgbtn, lv_imgbtn_state_t state)`

Get the right image in a given state

参数

- **imgbtn** -- pointer to an image button object
- **state** -- the state where to get the image (from `lv_btn_state_t`)

返回 pointer to the left image source (a C array or path to a file)

Variables

```
const lv_obj_class_t lv_imgbtn_class
struct lv_imgbtn_t
```

Public Members

```
lv_obj_t obj
const void *img_src_mid[LV_IMGBTN_STATE_NUM]
const void *img_src_left[LV_IMGBTN_STATE_NUM]
const void *img_src_right[LV_IMGBTN_STATE_NUM]
lv_img_cf_t act_cf
```

6.3.5 Keyboard (lv_keyboard)

Overview

The Keyboard object is a special *Button matrix* with predefined keymaps and other features to realize a virtual keyboard to write texts into a *Text area*.

Parts and Styles

Similarly to Button matrices Keyboards consist of 2 part:

- **LV_PART_MAIN** The main part. Uses all the typical background properties
- **LV_PART_ITEMS** The buttons. Also uses all typical background properties as well as the *text* properties.

Usage

Modes

The Keyboards have the following modes:

- **LV_KEYBOARD_MODE_TEXT_LOWER** Display lower case letters
- **LV_KEYBOARD_MODE_TEXT_UPPER** Display upper case letters
- **LV_KEYBOARD_MODE_TEXT_SPECIAL** Display special characters
- **LV_KEYBOARD_MODE_NUMBER** Display numbers, +/- sign, and decimal dot.

The TEXT modes' layout contains buttons to change mode.

To set the mode manually, use `lv_keyboard_set_mode(kb, mode)`. The default mode is `LV_KEYBOARD_MODE_TEXT_UPPER`.

Assign Text area

You can assign a [Text area](#) to the Keyboard to automatically put the clicked characters there. To assign the text area, use `lv_keyboard_set_textarea(kb, ta)`.

New Keymap

You can specify a new map (layout) for the keyboard with `lv_keyboard_set_map(kb, map)` and `lv_keyboard_set_ctrl_map(kb, ctrl_map)`. Learn more about the [Button matrix](#) object. Keep in mind that using following keywords will have the same effect as with the original map:

- `LV_SYMBOL_OK` Apply.
- `LV_SYMBOL_CLOSE` or `LV_SYMBOL_KEYBOARD CLOSE` Close.
- `LV_SYMBOL_BACKSPACE` Delete on the left.
- `LV_SYMBOL_LEFT` Move the cursor left.
- `LV_SYMBOL_RIGHT` Move the cursor right.
- `LV_SYMBOL_NEW_LINE` New line.
- "ABC" Load the uppercase map.
- "abc" Load the lower case map.
- "I#" Load the lower case map.

Events

- `LV_EVENT_VALUE_CHANGED` Sent when the button is pressed/released or repeated after long press. The event data is set to the ID of the pressed/released button.
- `LV_EVENT_READY` - The *Ok* button is clicked.
- `LV_EVENT_CANCEL` - The *Close* button is clicked.

The keyboard has a **default event handler** callback called `lv_keyboard_def_event_cb`, which handles the button pressing, map changing, the assigned text area, etc. You can remove it and replace it with a custom event handler if you wish.

注解: In 8.0 and newer, adding an event handler to the keyboard does not remove the default event handler. This behavior differs from v7, where adding an event handler would always replace the previous one.

Learn more about [Events](#).

Keys

- LV_KEY_RIGHT/UP/LEFT/RIGHT To navigate among the buttons and select one.
- LV_KEY_ENTER To press/release the selected button.

Learn more about [Keys](#).

Examples

C

Keyboard with text area

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_keyboard/lv_ex_keyboard_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_keyboard/lv_ex_keyboard_1.py
```

MicroPython

Keyboard with text area

No examples yet.

API

TypeDefs

```
typedef uint8_t lv_keyboard_mode_t
```

Enums

enum [anonymous]

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
```

Functions

`lv_obj_t *lv_keyboard_create(lv_obj_t *parent)`

Create a keyboard objects

参数 **par** -- pointer to an object, it will be the parent of the new keyboard

返回 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.

参数

- **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)

参数

- **kb** -- pointer to a Keyboard object
- **mode** -- the mode from 'lv_keyboard_mode_t'

`void lv_keyboard_set_map(lv_obj_t *kb, lv_keyboard_mode_t mode, const char *map[], const lv_btnmatrix_ctrl_t ctrl_map[])`

Set a new map for the keyboard

参数

- **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_btnmatrix_set_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.

参数 **kb** -- pointer to a Keyboard object

返回 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)

参数 **kb** -- pointer to a Keyboard object

返回 the current mode from 'lv_keyboard_mode_t'

`static inline const char **lv_keyboard_get_map_array(const lv_obj_t *kb)`

Get the current map of a keyboard

参数 **kb** -- pointer to a keyboard object

返回 the current map

`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 be called from it to handle the button clicks

参数

- **kb** -- pointer to a keyboard
- **event** -- the triggering event

Variables

```
const lv_obj_class_t lv_keyboard_class
struct lv_keyboard_t
```

Public Members

```
lv_btnmatrix_t btonm
lv_obj_t *ta
lv_keyboard_mode_t mode
```

6.3.6 LED (**lv_led**)

Overview

The LEDs are rectangle-like (or circle) object whose brightness can be adjusted. With lower brightness the colors of the LED become darker.

Parts and Styles

The LEDs have only one main part, called **LV_LED_PART_MAIN** and it uses all the typical background style properties.

Usage

Color

You can set the color of the LED with `lv_led_set_color(led, lv_color_hex(0xff0080))`. This will be used as background color, border color, and shadow color.

Brightness

You can set their brightness with `lv_led_set_bright(led, bright)`. The brightness should be between 0 (darkest) and 255 (lightest).

Toggle

Use `lv_led_on(led)` and `lv_led_off(led)` to set the brightness to a predefined ON or OFF value. The `lv_led_toggle(led)` toggles between the ON and OFF state.

Events

- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` is sent for the following types:
 - `LV_LED_DRAW_PART_RECTANGLE` The main rectangle. `LV_OBJ_DRAW_PART_RECTANGLE` is not sent by the base object.
 - * `part: LV_PART_MAIN`
 - * `rect_dsc`
 - * `draw_area`: the area of the rectangle

See the events of the *Base object* too.

Learn more about [Events](#).

Keys

No *Keys* are processed by the object type.

Learn more about [Keys](#).

Example

C

LED with custom style

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↳ lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_led/lv_ex_led_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↳ lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_led/lv_ex_led_1.py
```

MicroPython

No examples yet.

API

Functions

`lv_obj_t *lv_led_create(lv_obj_t *parent)`

Create a led objects

参数 **par** -- pointer to an object, it will be the parent of the new led

返回 pointer to the created led

`void lv_led_set_color(lv_obj_t *led, lv_color_t color)`

Set the color of the LED

参数

- **led** -- pointer to a LED object
- **color** -- the color of the the LED

`void lv_led_set_brightness(lv_obj_t *led, uint8_t bright)`

Set the brightness of a LED object

参数

- **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

参数 **led** -- pointer to a LED object

`void lv_led_off(lv_obj_t *led)`

Light off a LED

参数 **led** -- pointer to a LED object

`void lv_led_toggle(lv_obj_t *led)`

Toggle the state of a LED

参数 **led** -- pointer to a LED object

`uint8_t lv_led_get_brightness(const lv_obj_t *obj)`

Get the brightness of a LED object

参数 **led** -- pointer to LED object

返回 bright 0 (max. dark) ... 255 (max. light)

Variables

`const lv_obj_class_t lv_led_class`

`struct lv_led_t`

Public Members

lv_obj_t **obj**

lv_color_t **color**

uint8_t **bright**

Current brightness of the LED (0..255)

6.3.7 List (*lv_list*)

Overview

The List is basically a rectangle with vertical layout to which Buttons and Texts can be added

Parts and Styles

Background

- **LV_PART_MAIN** The main part of the list that uses all the typical background properties
- **LV_PART_SCROLLBAR** The scrollbar. See the *Base objects* documentation for details.

Buttons and Texts See the *Button*'s and *Label*'s documentation.

Usage

Buttons

`lv_list_add_btn(list, icon, text)` adds a full-width button with an icon - that can be an image or symbol - and a text.

The text starts to scroll horizontally if its too long.

Texts

`lv_list_add_text(list, icon, text)` adds a text.

Events

No special events are sent by the List, but sent by the Button as usual.

Learn more about *Events*.

Keys

No *Keys* are processed by the object type.

Learn more about *Keys*.

Example

C

Simple List

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_list/lv_ex_list_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_list/lv_ex_list_1.py
```

MicroPython

No examples yet.

API

Functions

`lv_obj_t *lv_list_create(lv_obj_t *parent)`

`lv_obj_t *lv_list_add_text(lv_obj_t *list, const char *txt)`

`lv_obj_t *lv_list_add_btn(lv_obj_t *list, const char *icon, const char *txt)`

`const char *lv_list_get_btn_text(lv_obj_t *list, lv_obj_t *btn)`

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_btn_class`

6.3.8 Meter (lv_meter)

Overview

The Meter widget can visualize data in very flexible ways. It can show arcs, needles, ticks lines and labels.

Parts and Styles

- **LV_PART_MAIN** The background of the Meter. Uses the typical background properties.
- **LV_PART_TICK** The tick lines and labels using the *line* and *text* style properties.
- **LV_PART_INDICATOR** The needle line or image using the *line* and *img* style properties, as well as the background properties to draw a square (or circle) on the pivot of the needles. Padding makes the square larger.
- **LV_PART_ITEMS** The arcs using the *arc* properties.

Usage

Add a scale

First a *Scale* needs to be added to the Meter with `lv_meter_scale_t * scale = lv_meter_add_scale(meter)`. The Scale has minor and major ticks and labels on the major ticks. Later indicators (needles, arcs, tick modifiers) can be added to the meter

Any number of scales can be added to Meter.

The minor tick lines can be configured with: `lv_meter_set_scale_ticks(meter, scale, tick_count, line_width, tick_length, ctick_olor)`.

To add major tick lines use `lv_meter_set_scale_major_ticks(meter, scale, nth_major, tick_width, tick_length, tick_color, label_gap)`. *nth_major* to specify how many minor ticks to skip to draw a major tick.

Labels are added automatically on major ticks with *label_gap* distance from the ticks with text proportionally to the values of the tick line.

`lv_meter_set_scale_range(meter, scale, min, max, angle_range, rotation)` sets the value and angle range of the scale.

Add indicators

Indicators need to be added to a Scale and their value is interpreted in the range of the Scale.

All the indicator add functions return `lv_meter_indicator_t *`.

Needle line

```
indic = lv_meter_add_needle_line(meter, scale, line_width, line_color, r_mod)
```

adds a needle line to a Scale. By default the length of the line is the same as the scale's radius but `r_mod` changes the length.

`lv_meter_set_indicator_value(meter, indic, value)` sets the value of the indicator.

Needle image

```
indic = lv_meter_add_needle_img(meter, scale, img_src, pivot_x, pivot_y)
```

sets an image that will be used as a needle. `img_src` should be a needle pointing to the right like this -0--->. `pivot_x` and `pivot_y` sets the pivot point of the rotation relative to the top left corner of the image.

`lv_meter_set_indicator_value(meter, inidicator, value)` sets the value of the indicator.

Arc

```
indic = lv_meter_add_arc(meter, scale, arc_width, arc_color, r_mod)
```

adds and arc indicator. . By default the radius of the arc is the same as the scale's radius but `r_mod` changes the radius.

`lv_meter_set_indicator_start_value(meter, indic, value)` and
`lv_meter_set_indicator_end_value(meter, inidicator, value)` sets the value of the indicator.

Scale lines (ticks)

```
indic = lv_meter_add_scale_lines(meter, scale, color_start, color_end, local, width_mod)
```

adds an indicator that modifies the ticks lines. If `local` is `true` the ticks' color will be faded from `color_start` to `color_end` in the indicator's start and end value range. If `local` is `false` `color_start` and `color_end` will be mapped to the start and end value of the scale and only a "slice" of that color gradient will be visible in the indicator's start and end value range. `width_mod` modifies the width of the tick lines.

`lv_meter_set_indicator_start_value(meter, inidicator, value)` and
`lv_meter_set_indicator_end_value(meter, inidicator, value)` sets the value of the indicator.

Events

- `LV_EVENT_DRAW_PART_BEGIN` and `LV_EVENT_DRAW_PART_END` is sent for the following types:
 - `LV_METER_DRAW_PART_ARC` The arc indicator
 - * `part`: `LV_PART_ITEMS`
 - * `sub_part_ptr`: pointer to the indicator
 - * `arc_dsc`
 - * `radius`: radius of the arc
 - * `p1`: center of the arc
 - `LV_METER_DRAW_PART_NEEDLE_LINE` The needle lines
 - * `part`: `LV_PART_ITEMS`

- * `p1, p2` points of the line
- * `line_dsc`
- * `sub_part_ptr`: pointer to the indicator
- `LV_METER_DRAW_PART_NEEDLE_IMG` The needle images
 - * `part`: `LV_PART_ITEMS`
 - * `p1, p2` points of the line
 - * `img_dsc`
 - * `sub_part_ptr`: pointer to the indicator
- `LV_METER_DRAW_PART_TICK` The tick lines and labels
 - * `part`: `LV_PART_TICKS`
 - * `value`: the value of the line
 - * `text`: `value` converted to decimal or `NULL` on minor lines
 - * `label_dsc`: label draw descriptor or `NULL` on minor lines
 - * `line_dsc`:
 - * `id`: the index of the line

See the events of the [Base object](#) too.

Learn more about [Events](#).

Keys

No keys are handled by the Meter widget.

Learn more about [Keys](#).

Example

C

Simple meter

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_meter/lv_ex_meter_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_meter/lv_ex_meter_1.py
```

MicroPython

No examples yet.

API

Enums

enum **lv_meter_indicator_type_t**

Values:

- enumerator **LV_METER_INDICATOR_TYPE_NEEDLE_IMG**
- enumerator **LV_METER_INDICATOR_TYPE_NEEDLE_LINE**
- enumerator **LV_METER_INDICATOR_TYPE_SCALE_LINES**
- enumerator **LV_METER_INDICATOR_TYPE_ARC**

Functions

*lv_obj_t *lv_meter_create(lv_obj_t *parent)*

Create a meter objects

参数 **parent** -- pointer to an object, it will be the parent of the new bar.

返回 pointer to the created meter

*lv_meter_scale_t *lv_meter_add_scale(lv_obj_t *obj)*

Add a new scale to the meter.

注解: Indicators can be attached to scales.

参数 **obj** -- pointer to a meter object

返回 the new scale

*void lv_meter_set_scale_ticks(lv_obj_t *obj, lv_meter_scale_t *scale, uint16_t cnt, uint16_t width, uint16_t len, lv_color_t color)*

Set the properties of the ticks of a scale

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to **meter**)
- **cnt** -- number of tick lines
- **width** -- width of tick lines
- **len** -- length of tick lines
- **color** -- color of tick lines

```
void lv_meter_set_scale_major_ticks(lv_obj_t *obj, lv_meter_scale_t *scale, uint16_t nth, uint16_t
width, uint16_t len, lv_color_t color, int16_t label_gap)
```

Make some "normal" ticks major ticks and set their attributes. Texts with the current value are also added to the major ticks.

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to **meter**)
- **nth** -- make every Nth normal tick major tick. (start from the first on the left)
- **width** -- width of the major ticks
- **len** -- length of the major ticks
- **color** -- color of the major ticks
- **label_gap** -- gap between the major ticks and the labels

```
void lv_meter_set_scale_range(lv_obj_t *obj, lv_meter_scale_t *scale, int32_t min, int32_t max, uint32_t
angle_range, uint32_t rotation)
```

Set the value and angular range of a scale.

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to **meter**)
- **min** -- the minimum value
- **max** -- the maximal value
- **angle_range** -- the angular range of the scale
- **rotation** -- the angular offset from the 3 o'clock position (clock-wise)

```
lv_meter_indicator_t *lv_meter_add_needle_line(lv_obj_t *obj, lv_meter_scale_t *scale, uint16_t width,
lv_color_t color, int16_t r_mod)
```

Add a needle line indicator the scale

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to **meter**)
- **width** -- width of the line
- **color** -- color of the line
- **r_mod** -- the radius modifier (added to the scale's radius) to get the lines length

返回

the new indicator

```
lv_meter_indicator_t *lv_meter_add_needle_img(lv_obj_t *obj, lv_meter_scale_t *scale, const void *src,
lv_coord_t pivot_x, lv_coord_t pivot_y)
```

Add a needle image indicator the scale

注解: the needle image should point to the right, like -O-->

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to **meter**)
- **src** -- the image source of the indicator. path or pointer to *lv_img_dsc_t*
- **pivot_x** -- the X pivot point of the needle
- **pivot_y** -- the Y pivot point of the needle

返回 the new indicator

```
lv_meter_indicator_t *lv_meter_add_arc(lv_obj_t *obj, lv_meter_scale_t *scale, uint16_t width, lv_color_t color,
                                         int16_t r_mod)
```

Add an arc indicator the scale

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to **meter**)
- **width** -- width of the arc
- **color** -- color of the arc
- **r_mod** -- the radius modifier (added to the scale's radius) to get the outer radius of the arc

返回 the new indicator

```
lv_meter_indicator_t *lv_meter_add_scale_lines(lv_obj_t *obj, lv_meter_scale_t *scale, lv_color_t
                                                color_start, lv_color_t color_end, bool local, int16_t
                                                width_mod)
```

Add a scale line indicator the scale. It will modify the ticks.

参数

- **obj** -- pointer to a meter object
- **scale** -- pointer to scale (added to **meter**)
- **color_start** -- the start color
- **color_end** -- the end color
- **local** -- tell how to map start and end color. true: the indicator's start and end_value; false: the scale's min max value
- **width_mod** -- add this the affected tick's width

返回 the new indicator

```
void lv_meter_set_indicator_value(lv_obj_t *obj, lv_meter_indicator_t *indic, int32_t value)
```

Set the value of the indicator. It will set start and and value to the same value

参数

- **obj** -- pointer to a meter object
- **indic** -- pointer to an indicator
- **value** -- the new value

```
void lv_meter_set_indicator_start_value(lv_obj_t *obj, lv_meter_indicator_t *indic, int32_t value)
```

Set the start value of the indicator.

参数

- **obj** -- pointer to a meter object
- **indic** -- pointer to an indicator
- **value** -- the new value

void lv_meter_set_indicator_end_value(*lv_obj_t* *obj, *lv_meter_indicator_t* *indic, int32_t value)
Set the start value of the indicator.

参数

- **obj** -- pointer to a meter object
- **indic** -- pointer to an indicator
- **value** -- the new value

Variables

```
const lv_obj_class_t lv_meter_class
struct lv_meter_scale_t
```

Public Members

```
lv_color_t tick_color
uint16_t tick_cnt
uint16_t tick_length
uint16_t tick_width
lv_color_t tick_major_color
uint16_t tick_major_nth
uint16_t tick_major_length
uint16_t tick_major_width
int16_t label_gap
int16_t label_color
int32_t min
int32_t max
int16_t r_mod
uint16_t angle_range
int16_t rotation
struct lv_meter_indicator_t
```

Public Members

```

lv_meter_scale_t *scale
lv_meter_indicator_type_t type
lv_opa_t opa
int32_t start_value
int32_t end_value
const void *src
lv_point_t pivot
struct lv_meter_indicator_t::[anonymous]::[anonymous] needle_img
uint16_t width
int16_t r_mod
lv_color_t color
struct lv_meter_indicator_t::[anonymous]::[anonymous] needle_line
struct lv_meter_indicator_t::[anonymous]::[anonymous] arc
int16_t width_mod
lv_color_t color_start
lv_color_t color_end
uint8_t local_grad
struct lv_meter_indicator_t::[anonymous]::[anonymous] scale_lines
union lv_meter_indicator_t::[anonymous] type_data

```

struct **lv_meter_t**

Public Members

```

lv_obj_t obj
lv_ll_t scale_ll
lv_ll_t indicator_ll

```

6.3.9 Message box (lv_msgbox)**Overview**

The Message boxes act as pop-ups. They are built from a background container, a title, an optional close button, a text and optional buttons.

The text will be broken into multiple lines automatically and the height will be set automatically to include the text and the buttons.

The message box can be modal (blocking clicks on the rest of the screen) or not modal.

Parts and Styles

The mesage box is built from other widgets so you can check these widget's documentation for details.

- Background: `lv_obj`
- Close button: `lv_btn`
- Title and text: `lv_label`
- Buttons: `lv_btncmatrix`

Usage

Create a message box

`lv_msgbox_create(parent, title, txt, btn_txts[], add_close_btn)` creates a message box.

If `parent` is `NULL` the message box will be modal. `title` and `txt` are strings for the title and the text. `btn_txts[]` is an array with the buttons' text. E.g. `const char * btn_txts[] = {"OK", "Cancel", NULL};`. `add_colse_btn` can be `true` or `false` to add/don't add a close button.

Get the parts

The building blocks of the message box can be obtained using the following functions:

```
lv_obj_t * lv_msgbox_get_title(lv_obj_t * mbox);
lv_obj_t * lv_msgbox_get_close_btn(lv_obj_t * mbox);
lv_obj_t * lv_msgbox_get_text(lv_obj_t * mbox);
lv_obj_t * lv_msgbox_get_btns(lv_obj_t * mbox);
```

Close the message box

`lv_msgbox_close(msgbox)` closes (deletes) the message box.

Events

- `LV_EVENT_VALUE_CHANGED` is sent by the buttons if one of them is clicked. `LV_OBJ_FLAG_EVENT_BUBBLE` is enabled on the buttons so you can add events to the message box itself. In the event handler, `lv_event_get_target(e)` will return the button matrix and `lv_event_get_current_target(e)` will givreturn the message box. `lv_msgbox_get_active_btn_text(msgbox)` can be used to get the text of the clicked button.

Learn more about [Events](#).

Keys

Keys have effect on the close button and button matrix. You can add them manually to a group if required.

Learn more about [Keys](#).

Example

C

Simple Message box

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_msgbox/lv_ex_msgbox_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_msgbox/lv_ex_msgbox_1.py
```

Modal

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_msgbox/lv_ex_msgbox_2.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_msgbox/lv_ex_msgbox_2.py
```

MicroPython

No examples yet.

API

Functions

```
lv_obj_t *lv_msgbox_create(lv_obj_t *parent, const char *title, const char *txt, const char *btn_txts[], bool add_close_btn)
```

Create a message box objects

参数

- **parent** -- pointer to parent or NULL to create a full screen modal message box
- **title** -- the title of the message box
- **txt** -- the text of the message box
- **btn_txts** -- the buttons as an array of texts terminated by an "" element. E.g. {"btn1", "btn2", ""}
- **add_close_btn** -- true: add a close button

返回 pointer to the message box object

```

lv_obj_t *lv_msgbox_get_title(lv_obj_t *mbox)

lv_obj_t *lv_msgbox_get_close_btn(lv_obj_t *mbox)

lv_obj_t *lv_msgbox_get_text(lv_obj_t *mbox)

lv_obj_t *lv_msgbox_get_btns(lv_obj_t *mbox)

const char *lv_msgbox_get_active_btn_text(lv_obj_t *mbox)

void lv_msgbox_close(lv_obj_t *mbox)

```

Variables

const lv_obj_class_t **lv_msgbox_class**

6.3.10 Span (lv_span)

Overview

A spangroup is the object that is used to display rich text. Different from the label object, **spangroup** can automatically organize text of different fonts, colors, and sizes into the spangroup obj.

Parts and Styles

- **LV_PART_MAIN** The spangroup has only one part.

Usage

Set text and style

The spangroup object uses span to describe text and text style. so, first we need to create span descriptor using `lv_span_t * span = lv_spangroup_new_span(spangroup)`. Then use `lv_span_set_text(span, "text")` to set text. The style of the modified text is the same as the normal style used, eg:`lv_style_set_text_color(&span->style, lv_palette_main(LV_PALETTE_RED))`.

If spangroup object `mode != LV_SPAN_MODE_FIXED` you must call `lv_spangroup_refr_mode()` after you have modified span style(eg:set text, changed the font size, del span).

Retreiving a span child

Spangroups store their children differently from normal objects, so normal functions for getting children won't work.

`lv_spangroup_get_child(spangroup, id)` will return a pointer to the child span at index `id`. In addition, `id` can be negative to index from the end of the spangroup where -1 is the youngest child, -2 is second youngest, etc.

e.g. `lv_span_t* span = lv_spangroup_get_child(spangroup, 0)` will return the first child of the spangroup. `lv_span_t* span = lv_spangroup_get_child(spangroup, -1)` will return the last (or most recent) child.

Child Count

Use the function `lv_spangroup_get_child_cnt(spangroup)` to get back the number of spans the group is maintaining.

e.g. `uint32_t size = lv_spangroup_get_child_cnt(spangroup)`

Text align

like label object, the spangroup can be set to one the following modes:

- `LV_TEXT_ALIGN_LEFT` Align text to left.
- `LV_TEXT_ALIGN_CENTER` Align text to center.
- `LV_TEXT_ALIGN_RIGHT` Align text to right.
- `LV_TEXT_ALIGN_AUTO` Align text auto.

use function `lv_spangroup_set_align(spangroup, LV_TEXT_ALIGN_CENTER)` to set text align.

Modes

The spangroup can be set to one the following modes:

- `LV_SPAN_MODE_FIXED` fixes the object size.
- `LV_SPAN_MODE_EXPAND` Expand the object size to the text size but stay on a single line.
- `LV_SPAN_MODE_BREAK` Keep width, break the too long lines and auto expand height.

Use `lv_spangroup_set_mode(spangroup, LV_SPAN_MODE_BREAK)` to set object mode.

Overflow

The spangroup can be set to one the following modes:

- `LV_SPAN_OVERFLOW_CLIP` truncates the text at the limit of the area.
- `LV_SPAN_OVERFLOW_ELLIPSIS` will display an ellipsis(....) when text overflows the area.

Use `lv_spangroup_set_overflow(spangroup, LV_SPAN_OVERFLOW_CLIP)` to set object overflow mode.

first line indent

Use `lv_spangroup_set_indent(spangroup, 20)` to set the indent of the first line, in pixels.

Events

No special events are sent by this widget.

Learn more about [Events](#).

Keys

No *Keys* are processed by the object type.

Learn more about [Keys](#).

Example

C

span with custom style

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪ lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_span/lv_ex_span_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪ lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_span/lv_ex_span_1.py
```

MicroPython

No examples yet.

API

Typedefs

```
typedef uint8_t lv_span_overflow_t
```

```
typedef uint8_t lv_span_mode_t
```

Enums

enum [anonymous]

Values:

enumerator **LV_SPAN_OVERFLOW_CLIP**

enumerator **LV_SPAN_OVERFLOW_ELLIPSIS**

enum [anonymous]

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

Functions

lv_obj_t ***lv_spangroup_create**(*lv_obj_t* *par)

Create a spangroup objects

参数 **par** -- pointer to an object, it will be the parent of the new spangroup

返回 pointer to the created spangroup

lv_span_t ***lv_span_create**(*lv_obj_t* *obj)

Create a span string descriptor and add to spangroup.

参数 **obj** -- pointer to a spangroup object.

返回 pointer to the created span.

void **lv_span_del**(*lv_obj_t* *obj, *lv_span_t* *span)

Remove the span from the spangroup and free memory.

参数

- **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.

参数

- **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.

参数

- **span** -- pointer to a span.

- **text** -- pointer to a text.

void lv_span_set_align(*lv_obj_t* *obj, lv_text_align_t align)
Set the align of the spangroup.

参数

- **obj** -- pointer to a spangroup object.
- **align** -- see lv_text_align_t for details.

void lv_span_set_overflow(*lv_obj_t* *obj, *lv_span_overflow_t* overflow)
Set the overflow of the spangroup.

参数

- **obj** -- pointer to a spangroup object.
- **overflow** -- see lv_span_overflow_t for details.

void lv_span_set_indent(*lv_obj_t* *obj, lv_coord_t indent)
Set the indent of the spangroup.

参数

- **obj** -- pointer to a spangroup object.
- **indent** -- The first line indentation

void lv_span_set_mode(*lv_obj_t* *obj, *lv_span_mode_t* mode)
Set the mode of the spangroup.

参数

- **obj** -- pointer to a spangroup object.
- **mode** -- see lv_span_mode_t for details.

lv_text_align_t **lv_span_get_align(*lv_obj_t* *obj)**
get the align of the spangroup.

参数 **obj** -- pointer to a spangroup object.

返回 the align value.

lv_span_overflow_t **lv_span_get_overflow(*lv_obj_t* *obj)**
get the overflow of the spangroup.

参数 **obj** -- pointer to a spangroup object.

返回 the overflow value.

lv_coord_t **lv_span_get_indent(*lv_obj_t* *obj)**
get the indent of the spangroup.

参数 **obj** -- pointer to a spangroup object.

返回 the indent value.

lv_span_mode_t **lv_span_get_mode(*lv_obj_t* *obj)**
get the mode of the spangroup.

参数 **obj** -- pointer to a spangroup object.

lv_coord_t **lv_span_get_max_line_h(*lv_obj_t* *obj)**
get max line height of all span in the spangroup.

参数 **obj** -- pointer to a spangroup object.

lv_coord_t **lv_span_get_expand_width**(*lv_obj_t* *obj)
 get the width when all span of spangroup on a line. include spangroup pad.

参数 **obj** -- pointer to a spangroup object.

lv_coord_t **lv_span_get_expand_height**(*lv_obj_t* *obj, lv_coord_t width)
 get the height with width fixed. the height include spangroup pad.

参数 **obj** -- pointer to a spangroup object.

void **lv_span_refr_mode**(*lv_obj_t* *obj)
 update the mode of the spangroup.

参数 **obj** -- pointer to a spangroup object.

Variables

const lv_obj_class_t **lv_spangroup_class**
 struct **lv_span_t**

Public Members

char ***txt**
lv_style_t **style**
 uint8_t **static_flag**
 struct **lv_spangroup_t**
#include <lv_span.h> Data of label

Public Members

lv_obj_t **obj**
 lv_coord_t **indent**
 lv_ll_t **child_ll**
 uint8_t **mode**
 uint8_t **align**
 uint8_t **overflow**

6.3.11 Spinbox (lv_spinbox)

Overview

The Spinbox contains a number as text which can be increased or decreased by *Keys* or API functions. Under the hood the Spinbox is a modified *Text area*.

Parts and Styles

The parts of the Spinbox are identical to the *Text area*.

Value, range and step

`lv_spinbox_set_value(spinbox, 1234)` sets a new value on the Spinbox.

`lv_spinbox_increment(spinbox)` and `lv_spinbox_decrement(spinbox)` increments/decrements the value of the Spinbox according to the currently selected digit.

`lv_spinbox_set_range(spinbox, -1000, 2500)` sets a range. If the value is changed by `lv_spinbox_set_value`, by `lv_spinbox_increment/decrement` this range will be respected.

`lv_spinbox_set_step(spinbox, 100)` sets which digits to change on increment/decrement. Only multiples of ten can be set, and not for example 3.

`lv_spinbox_set_pos(spinbox, 1)` sets the cursor to a specific digit to change on increment/decrement. For example position '0' sets the cursor to the least significant digit.

Format

`lv_spinbox_set_digit_format(spinbox, digit_count, separator_position)` sets the number format. `digit_count` is the number of digits excluding the decimal separator and the sign. `separator_position` is the number of digits before the decimal point. If 0, no decimal point is displayed.

Rollover

`lv_spinbox_set_rollover(spinbox, true/false)` enables/disabled rollover mode. If either the minimum or maximum value is reached with rollover enabled, the value will change to the other limit. If rollover is disabled the value will remain at the minimum or maximum value.

Events

- `LV_EVENT_VALUE_CHANGED` Sent when the value has changed.

See the events of the *Text area* too.

Learn more about *Events*.

Keys

- LV_KEY_LEFT/RIGHT With *Keypad* move the cursor left/right. With *Encoder* decrement/increment the selected digit.
- LV_KEY_UP/DOWN With *Keypad* and *Encoder* increment/decrement the value.
- LV_KEY_ENTER With *Encoder* got the net digit. Jump to the first after the last.

Example

C

Simple Spinbox

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_spinbox/lv_ex_spinbox_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_spinbox/lv_ex_spinbox_1.py
```

MicroPython

No examples yet.

API

Functions

`lv_obj_t *lv_spinbox_create(lv_obj_t *parent)`

Create a spinbox objects

参数 **par** -- pointer to an object, it will be the parent of the new spinbox

返回 pointer to the created spinbox

`void lv_spinbox_set_rollover(lv_obj_t *obj, bool b)`

Set spinbox rollover function

参数

- **spinbox** -- pointer to spinbox
- **b** -- true or false to enable or disable (default)

`void lv_spinbox_set_value(lv_obj_t *obj, int32_t i)`

Set spinbox value

参数

- **spinbox** -- pointer to spinbox
- **i** -- value to be set

`void lv_spinbox_set_digit_format(lv_obj_t *obj, uint8_t digit_count, uint8_t separator_position)`

Set spinbox digit format (digit count and decimal format)

参数

- **spinbox** -- pointer to spinbox
- **digit_count** -- number of digit excluding the decimal separator and the sign
- **separator_position** -- 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

参数

- **spinbox** -- pointer to spinbox
- **step** -- steps on increment/decrement

void lv_spinbox_set_range(*lv_obj_t* *obj, int32_t range_min, int32_t range_max)

Set spinbox value range

参数

- **spinbox** -- pointer to spinbox
- **range_min** -- maximum value, inclusive
- **range_max** -- minimum value, inclusive

bool lv_spinbox_get_rollover(*lv_obj_t* *obj)

Get spinbox rollover function status

参数 **spinbox** -- 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)

参数 **spinbox** -- pointer to spinbox

返回 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)

参数 **spinbox** -- pointer to spinbox

返回 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

参数 **spinbox** -- pointer to spinbox

void lv_spinbox_step_prev(*lv_obj_t* *obj)

Select next higher digit for edition by multiplying the step by 10

参数 **spinbox** -- pointer to spinbox

void lv_spinbox_increment(*lv_obj_t* *obj)

Increment spinbox value by one step

参数 **spinbox** -- pointer to spinbox

void lv_spinbox_decrement(*lv_obj_t* *obj)

Decrement spinbox value by one step

参数 **spinbox** -- pointer to spinbox

Variables

```
const lv_obj_class_t lv_spinbox_class
struct lv_spinbox_t
```

Public Members

```
lv_textarea_t ta
int32_t value
int32_t range_max
int32_t range_min
int32_t step
uint16_t digit_count
uint16_t dec_point_pos
uint16_t rollover
```

Example

6.3.12 Spinner (*lv_spinner*)

Overview

The Spinner object is a spinning arc over a ring.

Parts and Styles

The parts are identical to the parts of *lv_arc*.

Usage

Create a spinner

To create a spinner use `lv_spinner_create(parent, spin_time, arc_length)`. `spin_time` sets the spin time in milliseconds, `arc_length` sets the length of the spinning arc in degrees.

Events

No special events are sent by the Spinner.

See the events of the [Arc](#) too.

Learn more about [Events](#).

Keys

No *Keys* are processed by the object type.

Learn more about [Keys](#).

Example

C

Simple spinner

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_spinner/lv_ex_spinner_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_spinner/lv_ex_spinner_1.py
```

MicroPython

API

Functions

`lv_obj_t *lv_spinner_create(lv_obj_t *parent, uint32_t time, uint32_t arc_length)`

6.3.13 Tabview (lv_tabview)

Overview

The Tab view object can be used to organize content in tabs. The Tab view is built from other widgets:

- Main container: `lv_obj`)
 - Tab buttons: `lv_btndmatrix`
 - Container for the tabs: `lv_obj`
 - * Content of the tabs: `lv_obj`

The tab buttons can be positioned on the top, bottom, left and right side of the Tab view.

A new tab can be selected either by clicking on a tab button or by sliding horizontally on the content.

Parts and Styles

There are no special parts on the Tab view but the `lv_obj` and `lv_bttnmatrix` widgets are used to create the Tab view.

Usage

Create a Tab view

`lv_tabview_create(parent, tab_pos, tab_size);` creates a new empty Tab view. `tab_pos` can be `LV_DIR_TOP/BOTTOM/LEFT/RIGHT` to position the tab buttons to a side. `tab_size` is the height (in case of `LV_DIR_TOP/BOTTOM`) or width (in case of `LV_DIR_LEFT/RIGHT`) tab buttons.

Add tabs

New tabs can be added with `lv_tabview_add_tab(tabview, "Tab name")`. This will return a pointer to an `lv_obj` object where the tab's content can be created.

Change tab

To select a new tab you can:

- Click on its tab button
- Slide horizontally
- Use `lv_tabview_set_act(tabview, id, LV_ANIM_ON/OFF)` function

Get the parts

`lv_tabview_get_content(tabview)` returns the container for the tabs, `lv_tabview_get_tab_btns(tabview)` returns the Tab buttons object which is a *Button matrix*.

Events

- `LV_EVENT_VALUE_CHANGED` Sent when a new tab is selected by sliding or clicking the tab button. `lv_tabview_get_tab_act(tabview)` returns the zero based index of the current tab.

Learn more about [Events](#).

Keys

Keys have effect only on the tab buttons (Button matrix). Add manually to a group if required.

Learn more about [Keys](#).

Example

C

Simple Tabview

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_tabview/lv_ex_tabview_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_tabview/lv_ex_tabview_1.py
```

MicroPython

No examples yet.

API

Functions

```
lv_obj_t *lv_tabview_create(lv_obj_t *parent, lv_dir_t tab_pos, lv_coord_t tab_size)
```

```
lv_obj_t *lv_tabview_add_tab(lv_obj_t *tv, const char *name)
```

```
lv_obj_t *lv_tabview_get_content(lv_obj_t *tv)
```

```
lv_obj_t *lv_tabview_get_tab_btns(lv_obj_t *tv)
```

```
void lv_tabview_set_act(lv_obj_t *obj, uint32_t id, lv_anim_enable_t anim_en)
```

```
uint16_t lv_tabview_get_tab_act(lv_obj_t *tv)
```

Variables

```
const lv_obj_class_t lv_tabview_class
```

```
struct lv_tabview_t
```

Public Members

```
lv_obj_t obj
char **map
uint16_t tab_cnt
uint16_t tab_cur
lv_dir_t tab_pos
```

6.3.14 Tile view (`lv_tileview`)

Overview

The Tile view is a container object whose elements (called *tiles*) can be arranged in grid form. By swiping the user can navigate between the tiles. Any direction of swiping can be disabled on the tiles individually to not allow moving from one tile to another.

If the Tile view is screen sized, the user interface resembles what you may have seen on smartwatches.

Parts and Styles

The Tile view is built from an `lv_obj` container and `lv_obj` tiles.

The parts and styles work the same as for `lv_obj`.

Usage

Add a tile

`lv_tileview_add_tile(tileview, row_id, col_id, dir)` creates a new tile on the `row_id`th row and `col_id`th column. `dir` can be `LV_DIR_LEFT/RIGHT/TOP/BOTTOM/HOR/VER/ALL` or OR-ed values to enable moving to the adjacent tiles into the given direction by swiping.

The returned value is an `lv_obj_t *` on which the content of the tab can be created.

Change tile

The Tile view can scroll to a tile with `lv_obj_set_tile(tileview, tile_obj, LV_ANIM_ON/OFF)` or `lv_obj_set_tile_id(tileview, col_id, row_id, LV_ANIM_ON/OFF)`;

Events

- **LV_EVENT_VALUE_CHANGED** Sent when a new tile loaded by scrolling.
`lv_tileview_get_tile_act(tabview)` can be used to get current tile.

Keys

Keys are not handled by the Tile view.

Learn more about [Keys](#).

Example

C

Tileview with content

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_tileview/lv_ex_tileview_1.c
```

```
Error encountered while trying to open /home/runner/work/100ask_lvgl_docs_8.x/100ask_
↪lvgl_docs_8.x/examples/lv_ex_widgets/lv_ex_tileview/lv_ex_tileview_1.py
```

MicroPython

No examples yet.

API

Functions

`lv_obj_t *lv_tileview_create(lv_obj_t *parent)`

Create a tileview objects

参数 **par** -- pointer to an object, it will be the parent of the new tileview

返回 pointer to the created tileview

`lv_obj_t *lv_tileview_add_tile(lv_obj_t *tv, uint8_t row_id, uint8_t col_id, lv_dir_t dir)`

`void lv_obj_set_tile(lv_obj_t *tv, lv_obj_t *tile_obj, lv_anim_enable_t anim_en)`

`void lv_obj_set_tile_id(lv_obj_t *tv, uint32_t col_id, uint32_t row_id, lv_anim_enable_t anim_en)`

Variables

```
const lv_obj_class_t lv_tileview_class
const lv_obj_class_t lv_tileview_tile_class
struct lv_tileview_t
```

Public Members

```
lv_obj_t obj
struct lv_tileview_tile_t
```

Public Members

```
lv_obj_t obj
lv_dir_t dir
```

6.3.15 Window (**lv_win**)

Overview

The Window is container-like object built from a header with title and buttons and a content area.

Parts and Styles

The Window is built from other widgets so you can check their documentation for details:

- Background: *lv_obj*
- Header on the background: *lv_obj*
- Title on the header: *lv_label*
- Buttons on the header: *lv_btn*
- Content area on the background: *lv_obj*

Usage

Create a Window

`lv_win_create(parent, header_height)` creates a Window with an empty header.

Title and buttons

Any number of texts (but typically only one) can be added to the header with `lv_win_add_title(win, "The title")`.

Control buttons can be added to the window's header with `lv_win_add_btn(win, icon, btn_width)`. `icon` can be any image source, and `btn_width` is the width of the button.

The title and the buttons will be added in the order the functions are called. So adding a button, a text and two other buttons will result in a button on the left, a title, and 2 buttons on the right. The width of the title is set to take all the remaining space on the header. In other words: it pushes to the right all the buttons that are added after the title.

Get the parts

`lv_win_get_header(win)` returns a pointer to the header, `lv_win_get_content(win)` returns a pointer to the content container to which the content of the window can be added.

Events

No special events are sent by the windows, however events can be added manually to the return value of `lv_win_add_btn`.

Learn more about [Events](#).

Keys

No *Keys* are handled by the window.

Learn more about [Keys](#).

Example

API

Functions

`lv_obj_t *lv_win_create(lv_obj_t *parent, lv_coord_t header_height)`

`lv_obj_t *lv_win_add_title(lv_obj_t *win, const char *txt)`

`lv_obj_t *lv_win_add_btn(lv_obj_t *win, const void *icon, lv_coord_t btn_w)`

`lv_obj_t *lv_win_get_header(lv_obj_t *win)`

`lv_obj_t *lv_win_get_content(lv_obj_t *win)`

Variables

```
const lv_obj_class_t lv_win_class
struct lv_win_t
```

Public Members

lv_obj_t **obj**

Layouts (布局)

7.1 Flex (弹性布局)

7.1.1 Overview (概述)

The Flexbox (or Flex for short) is a subset of [CSS Flexbox](#).

It can arrange items into rows or columns (tracks), handle wrapping, adjust the spacing between the items and tracks, handle *grow* to make the item(s) fill the remaining space with respect to min/max width and height.

To make an object flex container call `lv_obj_set_layout(obj, LV_LAYOUT_FLEX)`.

Note that the flex layout feature of LVGL needs to be globally enabled with `LV_USE_FLEX` in `lv_conf.h`.

Flexbox (或简称 Flex) 是 [CSS Flexbox](#) 的一个子集。

它可以将项目排列成行或列 (轨道)，处理环绕，调整项目和轨道之间的间距，处理 *grow* 以便项目填充剩余空间的最小/最大宽度和高度。

要使对象 flex 容器调用 `lv_obj_set_layout(obj, LV_LAYOUT_FLEX)`。

请注意，LVGL 的 flex 布局功能需要通过 `lv_conf.h` 中的 `LV_USE_FLEX` 全局启用。

7.1.2 Terms (约定)

- tracks: the rows or columns
- main direction: row or column, the direction in which the items are placed
- cross direction: perpendicular to the main direction
- wrap: if there is no more space in the track a new track is started
- grow: if set on an item it will grow to fill the remaining space on the track. The available space will be distributed among items respective to their grow value (larger value means more space)
- gap: the space between the rows and columns or the items on a track

- (tracks) 轨道: 行或列
- (main direction) 主要方向: 行或列, 物品放置的方向
- (cross direction) 横向: 垂直于主方向
- (wrap) 环绕: 如果曲目中没有更多空间, 则开始新曲目
- (grow) 增长: 如果设置在一个项目上, 它将增长以填充轨道上的剩余空间。可用空间将根据其增长值分配给各个项目 (值越大意味着空间越大)
- (gap) 间隙: 行和列或轨道上的项目之间的空间

7.1.3 Simple interface (简单接口)

With the following functions you can set a Flex layout on any parent.

使用以下功能, 您可以在任何父级上设置 Flex 布局。

Flex flow

`lv_obj_set_flex_flow(obj, flex_flow)`

The possible values for `flex_flow` are:

- `LV_FLEX_FLOW_ROW` Place the children in a row without wrapping
- `LV_FLEX_FLOW_COLUMN` Place the children in a column without wrapping
- `LV_FLEX_FLOW_ROW_WRAP` Place the children in a row with wrapping
- `LV_FLEX_FLOW_COLUMN_WRAP` Place the children in a column with wrapping
- `LV_FLEX_FLOW_ROW_REVERSE` Place the children in a row without wrapping but in reversed order
- `LV_FLEX_FLOW_COLUMN_REVERSE` Place the children in a column without wrapping but in reversed order
- `LV_FLEX_FLOW_ROW_WRAP_REVERSE` Place the children in a row without wrapping but in reversed order
- `LV_FLEX_FLOW_COLUMN_WRAP_REVERSE` Place the children in a column without wrapping but in reversed order

`lv_obj_set_flex_flow(obj, flex_flow)`

`flex_flow` 的可能值是:

- `LV_FLEX_FLOW_ROW` 将子元素排成一排而不包裹
- `LV_FLEX_FLOW_COLUMN` 将子项放在一列中而不换行
- `LV_FLEX_FLOW_ROW_WRAP` 将孩子排成一排并包裹起来
- `LV_FLEX_FLOW_COLUMN_WRAP` 将子元素放置在带有环绕的列中
- `LV_FLEX_FLOW_ROW_REVERSE` 将子元素排成一行而不换行, 但顺序相反
- `LV_FLEX_FLOW_COLUMN_REVERSE` 将子项放在一列中, 不换行, 但顺序相反
- `LV_FLEX_FLOW_ROW_WRAP_REVERSE` 将子元素排成一行而不换行, 但顺序相反
- `LV_FLEX_FLOW_COLUMN_WRAP_REVERSE` 将子项放在一列中, 不换行, 但顺序相反

Flex align

To manage the placement of the children use `lv_obj_set_flex_align(obj, main_place, cross_place, track_cross_place)`

- `main_place` determines how to distribute the items in their track on the main axis. E.g. flush the items to the right on `LV_FLOW_ROW_WRAP`. (It's called `justify-content` in CSS)
- `cross_place` determines how to distribute the items in their track on the cross axis. E.g. if the items have different height place them to the bottom of the track. (It's called `align-items` in CSS)
- `track_cross_place` determines how to distribute the tracks (It's called `align-content` in CSS)

要管理孩子的位置, 请使用 `lv_obj_set_flex_align(obj, main_place, cross_place, track_cross_place)`

- `main_place` 确定如何在主轴上的轨道中分布项目。例如。将“`LV_FLOW_ROW_WRAP`”上的项目向右刷新。(它在 CSS 中称为 `justify-content`)
- `cross_place` 确定如何在横轴上的轨道中分布项目。例如。如果项目具有不同的高度, 则将它们放置在轨道的底部。(在 CSS 中称为 `align-items`)
- `track_cross_place` 确定如何分配轨道 (在 CSS 中称为 `align-content`)

The possible values are:

- `LV_ALIGN_START` means left on a horizontally and top vertically. (default)
- `LV_ALIGN_END` means right on a horizontally and bottom vertically
- `LV_ALIGN_CENTER` simply center
- `LV_ALIGN_SPACE_EVENLY` items are distributed so that the spacing between any two items (and the space to the edges) is equal. Does not apply to `track_cross_place`.
- `LV_ALIGN_SPACE_AROUND` items are evenly distributed in the track with equal space around them. Note that visually the spaces aren't equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies. Not applies to `track_cross_place`.
- `LV_ALIGN_SPACE_BETWEEN` items are evenly distributed in the track: first item is on the start line, last item on the end line. Not applies to `track_cross_place`.

可能的值为:

- `LV_ALIGN_START` 表示在水平方向和垂直方向的顶部左侧。(默认)
- `LV_ALIGN_END` 表示水平和底部垂直
- `LV_ALIGN_CENTER` 只是居中
- `LV_ALIGN_SPACE_EVENLY` 项目是分布的, 因此任意两个项目之间的间距 (以及到边缘的空间) 是相等的。不适用于 `track_cross_place`。
- `LV_ALIGN_SPACE_AROUND` 项目均匀分布在轨道中, 它们周围的空间相等。请注意, 视觉上的空间并不相等, 因为所有项目的两侧都有相等的空间。第一个项目将与容器边缘有一个单位的空间, 但下一个项目之间有两个单位的空间, 因为下一个项目有自己的适用间距。不适用于 `track_cross_place`。
- `LV_ALIGN_SPACE_BETWEEN` 项目均匀分布在轨道中: 第一个项目在开始线上, 最后一个项目在结束线上。不适用于 `track_cross_place`。

Flex grow

Flex grow can be used to make one or more children fill the available space on the track. If more children has grow the available space will be distributed proportionally to the grow values. For example let's there is 400 px remaining space and 4 object with grow:

- A with grow = 1
- B with grow = 1
- C with grow = 2

A and B will have 100 px size, and C will have 200 px size.

Flex grow can be set on a child with `lv_obj_set_flex_flow(child, value)`. `value` needs to be > 1 or 0 to disable grow on the child.

Flex Growth 可用于让一个或多个孩子填充轨道上的可用空间。如果有更多的孩子成长，可用空间将与成长值成比例地分配。例如，让我们有 400 像素的剩余空间和 4 个增长的对象：

- A 增长 = 1
- B 增长 = 1
- C 增长 = 2

A 和 B 的大小为 100 px，而 C 的大小为 200 px。

可以使用 `lv_obj_set_flex_flow(child, value)` 在子节点上设置 Flex 增长。`value` 需要 > 1 或 0 禁用在孩子身上生长。

7.1.4 Style interface (样式接口)

All the Flex-related values are style properties under the hood and you can use them similarly to any other style property. The following flex related style properties exist:

- FLEX_FLOW
- FLEX_MAIN_PLACE
- FLEX_CROSS_PLACE
- FLEX_TRACK_PLACE
- FLEX_GROW

所有与 Flex 相关的值都是底层的样式属性，您可以像使用任何其他样式属性一样使用它们。存在以下与 flex 相关的样式属性：

- FLEX_FLOW
- FLEX_MAIN_PLACE
- FLEX_CROSS_PLACE
- FLEX_TRACK_PLACE
- FLEX_GROW

7.1.5 Other features (其它功能)

RTL

If the base direction of the container is set the `LV_BASE_DIRRTL` the meaning of `LV_FLEX_ALIGN_START` and `LV_FLEX_ALIGN_END` is swapped on `ROW` layouts. I.e. `START` will mean right.

The items on `ROW` layouts, and tracks of `COLUMN` layouts will be placed from right to left.

如果容器的基本方向设置为 `LV_BASE_DIRRTL`, `LV_FLEX_ALIGN_START` 和 `LV_FLEX_ALIGN_END` 的含义在 `ROW` 布局上交换。IE。`START` 表示正确。

`ROW` 布局上的项目和 `COLUMN` 布局的轨道将从右到左放置。

New track (新轨道)

You can force Flex to put an item into a new line with `lv_obj_add_flag(child, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK)`.

您可以使用 `lv_obj_add_flag(child, LV_OBJ_FLAG_FLEX_IN_NEW_TRACK)` 强制 Flex 将项目放入新行。

7.1.6 Example

7.1.7 API

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

LV_EXPORT_CONST_INT(LV_OBJ_FLAG_FLEX_IN_NEW_TRACK)

void **lv_flex_init**(void)

Initialize a flex layout the default values

参数 **flex** -- pointer to a flex layout descriptor

void **lv_obj_set_flex_flow**(*lv_obj_t* *obj, *lv_flex_flow_t* flow)

Set hot the item should flow

参数

- **flex** -- pointer to a flex layout descriptor
- **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 an tracks

参数

- **flex** -- pointer: to a flex layout descriptor
- **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_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

参数

- **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.

static inline void **lv_style_set_flex_flow**(*lv_style_t* *style, *lv_flex_flow_t* value)

static inline void **lv_style_set_flex_main_place**(*lv_style_t* *style, *lv_flex_align_t* value)

static inline void **lv_style_set_flex_cross_place**(*lv_style_t* *style, *lv_flex_align_t* value)

static inline void **lv_style_set_flex_track_place**(*lv_style_t* *style, *lv_flex_align_t* value)

static inline void **lv_style_set_flex_grow**(*lv_style_t* *style, uint8_t value)

```

static inline void lv_obj_set_style_flex_flow(lv_obj_t *obj, lv_flex_flow_t value, lv_style_selector_t selector)

static inline void lv_obj_set_style_flex_main_place(lv_obj_t *obj, lv_flex_align_t value,
                                                lv_style_selector_t selector)

static inline void lv_obj_set_style_flex_cross_place(lv_obj_t *obj, lv_flex_align_t value,
                                                    lv_style_selector_t selector)

static inline void lv_obj_set_style_flex_track_place(lv_obj_t *obj, lv_flex_align_t value,
                                                    lv_style_selector_t selector)

static inline void lv_obj_set_style_flex_grow(lv_obj_t *obj, uint8_t value, lv_style_selector_t selector)

static inline lv_flex_flow_t lv_obj_get_style_flex_flow(const lv_obj_t *obj, uint32_t part)

static inline lv_flex_align_t lv_obj_get_style_flex_main_place(const lv_obj_t *obj, uint32_t part)

static inline lv_flex_align_t lv_obj_get_style_flex_cross_place(const lv_obj_t *obj, uint32_t part)

static inline lv_flex_align_t lv_obj_get_style_flex_track_place(const lv_obj_t *obj, uint32_t part)

static inline uint8_t lv_obj_get_style_flex_grow(const lv_obj_t *obj, uint32_t part)

```

Variables

```

uint32_t LV_LAYOUT_FLEX  

lv_style_prop_t LV_STYLE_FLEX_FLOW  

lv_style_prop_t LV_STYLE_FLEX_MAIN_PLACE  

lv_style_prop_t LV_STYLE_FLEX_CROSS_PLACE  

lv_style_prop_t LV_STYLE_FLEX_TRACK_PLACE  

lv_style_prop_t LV_STYLE_FLEX_GROW

```

7.2 Grid (网格布局)

7.2.1 Overview (概述)

The Grid layout is a subset of [CSS Flexbox](#).

It can arrange items into 2D "table" that has rows or columns (tracks). The item can span through multiple columns or rows. The track's size can be set in pixel, to the largest item (**LV_GRID_CONTENT**) or in "Free unit" (FR) to distribute the free space proportionally.

To make an object a grid container call `lv_obj_set_layout(obj, LV_LAYOUT_GRID)`.

Note that the grid layout feature of LVGL needs to be globally enabled with `LV_USE_GRID` in `lv_conf.h`.

网格布局是 [CSS Flexbox](#) 的一个子集。

它可以将项目排列成具有行或列（轨道）的二维“表格”。该项目可以跨越多个列或行。轨道的大小可以设置为像素、最大项目（`LV_GRID_CONTENT`）或“空闲单元”（`FR`）以按比例分配空闲空间。

要使对象成为网格容器，请调用 `lv_obj_set_layout(obj, LV_LAYOUT_GRID)`。

请注意，LVGL 的网格布局功能需要通过 `lv_conf.h` 中的 `LV_USE_GRID` 全局启用。

7.2.2 Terms (约定)

- tracks: the rows or columns
- free unit (`FR`): if set on track's size is set in `FR` it will grow to fill the remaining space on the parent.
- gap: the space between the rows and columns or the items on a track
- 轨道：行或列
- 空闲单元（`FR`）：如果在 `FR` 中设置了轨道的大小，它将增长以填充父级上的剩余空间。
- 间隙：行和列或轨道上的项目之间的空间

7.2.3 Simple interface (简单的接口)

With the following functions you can easily set a Grid layout on any parent.

使用以下功能，您可以轻松地在任何父级上设置网格布局。

Grid descriptors

First you need to describe the size of rows and columns. It can be done by declaring 2 arrays and the track sizes in them. The last element must be `LV_GRID_TEMPLATE_LAST`.

For example:

首先，您需要描述行和列的大小。可以通过声明 2 个数组和其中的轨道大小来完成。最后一个元素必须是 `LV_GRID_TEMPLATE_LAST`。

例如：

```
static lv_coord_t column_dsc[] = {100, 400, LV_GRID_TEMPLATE_LAST}; /*2 columns
→with 100 and 400 px width*/
static lv_coord_t row_dsc[] = {100, 100, 100, LV_GRID_TEMPLATE_LAST}; /*3 100 px tall
→rows*/
```

To set the descriptors on a parent use `lv_obj_set_grid_dsc_array(obj, col_dsc, row_dsc)`.

Besides simple settings the size in pixel you can use two special values:

- `LV_GRID_CONTENT` set the width to the largest children on this track
- `LV_GRID_FR(X)` tell what portion of the remaining space should be used by this track. Larger value means larger space.

要在父级上设置描述符，请使用 `lv_obj_set_grid_dsc_array(obj, col_dsc, row_dsc)`。

除了简单的设置像素大小之外，您还可以使用两个特殊值：

- `LV_GRID_CONTENT` 将宽度设置为这条轨道上最大的孩子
- `LV_GRID_FR(X)` 告诉这个轨道应该使用剩余空间的哪一部分。更大的值意味着更大的空间。

Grid items (网格项)

By default the children are not added to the grid. They need to be added manually to a cell.

To do this call `lv_obj_set_grid_cell(child, column_align, column_pos, column_span, row_align, row_pos, row_span)`.

`column_align` and `row_align` determine how to align the children in its cell. The possible values are:

- `LV_GRID_ALIGN_START` means left on a horizontally and top vertically. (default)
- `LV_GRID_ALIGN_END` means right on a horizontally and bottom vertically
- `LV_GRID_ALIGN_CENTER` simply center

`column_pos` and `row_pos` means the zero based index of the cell into the item should be placed.

`column_span` and `row_span` means how many tracks should the item involve from the start cell. Must be > 1.

默认情况下，子项不会添加到网格中。它们需要手动添加到单元格中。

为此调用 `lv_obj_set_grid_cell(child, column_align, column_pos, column_span, row_align, row_pos, row_span)`。

`column_align` 和 `row_align` 确定如何在其单元格中对齐子项。可能的值为：

- `LV_GRID_ALIGN_START` 表示在水平方向和垂直方向的顶部左侧。(默认)
- `LV_GRID_ALIGN_END` 表示水平和底部垂直
- `LV_GRID_ALIGN_CENTER` 只是居中

`column_pos` 和 `row_pos` 表示应该放置项目中单元格的从零开始的索引。

`column_span` 和 `row_span` 表示该项目应该从起始单元格开始包含多少个轨道。必须 > 1.

Grid align (网格对齐)

If there are some empty space the track can be aligned several ways:

- `LV_GRID_ALIGN_START` means left on a horizontally and top vertically. (default)
- `LV_GRID_ALIGN_END` means right on a horizontally and bottom vertically
- `LV_GRID_ALIGN_CENTER` simply center
- `LV_GRID_ALIGN_SPACE_EVENLY` items are distributed so that the spacing between any two items (and the space to the edges) is equal. Not applies to `track_cross_place`.
- `LV_GRID_ALIGN_SPACE_AROUND` items are evenly distributed in the track with equal space around them. Note that visually the spaces aren't equal, since all the items have equal space on both sides. The first item will have one unit of space against the container edge, but two units of space between the next item because that next item has its own spacing that applies. Not applies to `track_cross_place`.
- `LV_GRID_ALIGN_SPACE_BETWEEN` items are evenly distributed in the track: first item is on the start line, last item on the end line. Not applies to `track_cross_place`.

To set the track's alignment use `lv_obj_set_grid_align(obj, column_align, row_align)`.

如果有一些空白空间，轨道可以通过几种方式对齐：

- `LV_GRID_ALIGN_START` 表示在水平方向和垂直方向的顶部左侧。(默认)
- `LV_GRID_ALIGN_END` 表示水平和底部垂直
- `LV_GRID_ALIGN_CENTER` 只是居中
- `LV_GRID_ALIGN_SPACE_EVENLY` 项目是分布的，因此任意两个项目之间的间距（以及到边缘的空间）是相等的。不适用于 `track_cross_place`。
- `LV_GRID_ALIGN_SPACE_AROUND` 项目均匀分布在轨道中，它们周围的空间相等。请注意，视觉上的空间并不相等，因为所有项目的两侧都有相等的空间。第一个项目将与容器边缘有一个单位的空间，但下一个项目之间有两个单位的空间，因为下一个项目有自己的适用间距。不适用于 `track_cross_place`。
- `LV_GRID_ALIGN_SPACE_BETWEEN` 项目均匀分布在轨道中：第一个项目在开始线上，最后一个项目在结束线上。不适用于 `track_cross_place`。

要设置轨道的对齐方式，请使用 `lv_obj_set_grid_align(obj, column_align, row_align)`。

7.2.4 Style interface (样式接口)

All the Grid related values are style properties under the hood and you can use them similarly to any other style properties. The following Grid related style properties exist:

- `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`

所有与 Grid 相关的值都是底层的样式属性，您可以像使用任何其他样式属性一样使用它们。存在以下与网格相关的样式属性：

- `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

7.2.5 Other features (其它功能)

RTL

If the base direction of the container is set to LV_BASE_DIRRTL, the meaning of LV_GRID_ALIGN_START and LV_GRID_ALIGN_END is swapped. I.e. START will mean right-most.

The columns will be placed from right to left.

如果容器的基本方向设置为LV_BASE_DIRRTL，则LV_GRID_ALIGN_START和LV_GRID_ALIGN_END的含义互换。IE。START表示最右边。

列将从右到左放置。

7.2.6 Example

7.2.7 API

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

LV_EXPORT_CONST_INT(LV_GRID_CONTENT)

LV_EXPORT_CONST_INT(LV_GRID_TEMPLATE_LAST)

void **lv_grid_init**(void)

void **lv_obj_set_grid_dsc_array**(*lv_obj_t* *obj, const lv_coord_t col_dsc[], const lv_coord_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(struct _lv_obj_t *obj, lv_grid_align_t column_align, uint8_t col_pos, uint8_t
                           col_span, lv_grid_align_t row_align, uint8_t row_pos, uint8_t row_span)
Set the cell of an object. The object's parent needs to have grid layout, else nothing will happen
```

参数

- **obj** -- pointer to an object
- **hor_place** -- the vertical alignment in the cell. LV_GRID_START/END/CENTER/STRETCH
- **col_pos** -- column ID
- **col_span** -- number of columns to take (≥ 1)
- **ver_place** -- the horizontal alignment in the cell. LV_GRID_START/END/CENTER/STRETCH
- **row_pos** -- row ID
- **row_span** -- number of rows to take (≥ 1)

```
static inline lv_coord_t lv_grid_fr(uint8_t x)
```

Just a wrapper to LV_GRID_FR for bindings.

```
static inline void lv_style_set_grid_row_dsc_array(lv_style_t *style, const lv_coord_t value[])
```

```
static inline void lv_style_set_grid_column_dsc_array(lv_style_t *style, const lv_coord_t value[])
```

```
static inline void lv_style_set_grid_row_align(lv_style_t *style, lv_grid_align_t value)
```

```
static inline void lv_style_set_grid_column_align(lv_style_t *style, lv_grid_align_t value)
```

```
static inline void lv_style_set_grid_cell_column_pos(lv_style_t *style, lv_coord_t value)
```

```
static inline void lv_style_set_grid_cell_column_span(lv_style_t *style, lv_coord_t value)
```

```
static inline void lv_style_set_grid_cell_row_pos(lv_style_t *style, lv_coord_t value)
```

```
static inline void lv_style_set_grid_cell_row_span(lv_style_t *style, lv_coord_t value)
```

```
static inline void lv_style_set_grid_cell_x_align(lv_style_t *style, lv_coord_t value)
```

```
static inline void lv_style_set_grid_cell_y_align(lv_style_t *style, lv_coord_t value)
```

```
static inline void lv_obj_set_style_grid_row_dsc_array(lv_obj_t *obj, const lv_coord_t value[],
                                                       lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_grid_column_dsc_array(lv_obj_t *obj, const lv_coord_t value[],
                                                        lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_grid_row_align(lv_obj_t *obj, lv_grid_align_t value,
                                                lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_grid_column_align(lv_obj_t *obj, lv_grid_align_t value,
                                                    lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_grid_cell_column_pos(lv_obj_t *obj, lv_coord_t value,
                                                       lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_grid_cell_column_span(lv_obj_t *obj, lv_coord_t value,
                                                       lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_grid_cell_row_pos(lv_obj_t *obj, lv_coord_t value,
                                                       lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_grid_cell_row_span(lv_obj_t *obj, lv_coord_t value,
                                                       lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_grid_cell_x_align(lv_obj_t *obj, lv_coord_t value,
                                                       lv_style_selector_t selector)
```

```
static inline void lv_obj_set_style_grid_cell_y_align(lv_obj_t *obj, lv_coord_t value,
                                                       lv_style_selector_t selector)
```

```
static inline const lv_coord_t *lv_obj_get_style_grid_row_dsc_array(const struct lv_obj_t *obj,
                                                                     uint32_t part)
```

```
static inline const lv_coord_t *lv_obj_get_style_grid_column_dsc_array(const struct lv_obj_t *obj,
                                                                     uint32_t part)
```

```
static inline lv_grid_align_t lv_obj_get_style_grid_row_align(const struct lv_obj_t *obj, uint32_t part)
```

```
static inline lv_grid_align_t lv_obj_get_style_grid_column_align(const struct lv_obj_t *obj, uint32_t
                                                               part)
```

```
static inline lv_coord_t lv_obj_get_style_grid_cell_column_pos(const struct lv_obj_t *obj, uint32_t
                                                               part)
```

```
static inline lv_coord_t lv_obj_get_style_grid_cell_column_span(const struct lv_obj_t *obj, uint32_t
                                                               part)
```

```
static inline lv_coord_t lv_obj_get_style_grid_cell_row_pos(const struct lv_obj_t *obj, uint32_t part)
```

```
static inline lv_coord_t lv_obj_get_style_grid_cell_row_span(const struct lv_obj_t *obj, uint32_t
                                                               part)
```

```
static inline lv_coord_t lv_obj_get_style_grid_cell_x_align(const struct lv_obj_t *obj, uint32_t part)
```

```
static inline lv_coord_t lv_obj_get_style_grid_cell_y_align(const struct lv_obj_t *obj, uint32_t part)
```

Variables

```
uint32_t LV_LAYOUT_GRID  
lv_style_prop_t LV_STYLE_GRID_COLUMN_DSC_ARRAY  
lv_style_prop_t LV_STYLE_GRID_COLUMN_ALIGN  
lv_style_prop_t LV_STYLE_GRID_ROW_DSC_ARRAY  
lv_style_prop_t LV_STYLE_GRID_ROW_ALIGN  
lv_style_prop_t LV_STYLE_GRID_CELL_COL_POS  
lv_style_prop_t LV_STYLE_GRID_CELL_COL_SPAN  
lv_style_prop_t LV_STYLE_GRID_CELL_X_ALIGN  
lv_style_prop_t LV_STYLE_GRID_CELL_ROW_POS  
lv_style_prop_t LV_STYLE_GRID_CELL_ROW_SPAN  
lv_style_prop_t LV_STYLE_GRID_CELL_Y_ALIGN
```

3rd party libraries(第三方库)

8.1 File System Interfaces(文件系统接口)

LVGL has a [File system](#) module to provides an abstraction layer for various file system drivers.

LVG has build in support for

- [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)

You still need to provide the drivers and libraries, this extensions provide only the bridge between FATFS, STDIO, POSIX, WIN32 and LVGL.

LVGL 有一个 [文件系统](#) 模块，为各种文件系统驱动程序提供一个抽象层。LVG 已经内置了对 [FATFS] 的支持 (http://elm-chan.org/fsw/ff/00index_e.html) - STDIO (Linux 和 Windows 使用 C 标准函数.eg fopen, fread) - POSIX (Linux 和 Windows 使用 POSIX 函数.eg open, read) - WIN32 (Windows 使用 Win32 API 函数。例如 CreateFileA、ReadFile) 您仍然需要提供驱动程序和库，此扩展仅提供 FATFS、STDIO、POSIX、WIN32 和 LVGL 之间的桥梁。

8.1.1 Usage(用法)

In `lv_conf.h` set a driver letter for one or more `LV_FS_USE_...` define(s). After that you can access files using that driver letter. Setting '`\0`' will disable use of that interface.

在 `lv_conf.h` 中为一个或多个 `LV_FS_USE_...` 定义设置一个驱动程序字母。之后，您可以使用该驱动程序号访问文件。设置'`\0`' 将禁止使用该接口。

8.2 BMP decoder(BMP 解码器)

This extension allows the use of BMP images in LVGL. This implementation uses `bmp-decoder` library. The pixels are read on demand (not the whole image is loaded) so using BMP images requires very little RAM.

If enabled in `lv_conf.h` by `LV_USE_BMP` LVGL will register a new image decoder automatically so BMP files can be directly used as image sources. For example:

```
lv_img_set_src(my_img, "S:/path/to/picture.bmp");
```

Note that, a file system driver needs to be registered to open images from files. Read more about it [here](#) or just enable one in `lv_conf.h` with `LV_USE_FS_...`

此扩展允许在 LVGL 中使用 BMP 图像。此实现使用 `bmp-decoder` 库。像素是按需读取的（不是整个图像被加载），因此使用 BMP 图像需要很少的 RAM。

如果在 `lv_conf.h` 中通过 `LV_USE_BMP` 启用 LVGL 将自动注册一个新的图像解码器，因此 BMP 文件可以直接用作图像源。例如：

```
lv_img_set_src(my_img, "S:/path/to/picture.bmp");
```

请注意，文件系统驱动程序需要注册才能从文件中打开图像。阅读更多相关信息 [此处](#) 或仅在 `lv_conf.h` 中使用 `LV_USE_FS_...` 启用一个

8.2.1 Limitations(限制条件)

- Only BMP files are supported and BMP images as C array (`lv_img_dsc_t`) are not. It's 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 file. If you want to store them in flash it's better to convert them to C array with LVGL's image converter.
- The BMP files color format needs to match with `LV_COLOR_DEPTH`. Use GIMP to save the image in the required format. Both RGB888 and ARGB888 works with `LV_COLOR_DEPTH 32`
- Palette is not supported.
- Because not the whole image is read in can not be zoomed or rotated.
- 仅支持 BMP 文件，不支持 BMP 图像作为 C 数组 (`lv_img_dsc_t`)。这是因为 BMP 文件和 LVGL 的图像格式存储图像数据的方式没有实际区别。
- BMP 文件只能从文件中加载。如果您想将它们存储在闪存中，最好使用 LVGL 的图像转换器 将它们转换为 C 数组。
- BMP 文件颜色格式需要与 `LV_COLOR_DEPTH` 匹配。使用 GIMP 以所需格式保存图像。RGB888 和 ARGB888 都适用于 `LV_COLOR_DEPTH 32`
- 不支持调色板。
- 因为不是整个图像被读入不能缩放或旋转。

8.2.2 Example

8.2.3 API

警告: doxygenfile: Cannot find file "lv_bmp.h"

8.3 JPG decoder(JPG 解码器)

Allow the use of JPG images in LVGL. Besides that it also allows the use of a custom format, called Split JPG (SJPG), which can be decided in more optimal way on embedded systems.

允许在 LVGL 中使用 JPG 图像。除此之外，它还允许使用称为 Split JPG (SJPG) 的自定义格式，可以在嵌入式系统上以更优化的方式决定。

8.3.1 Overview(概述)

- Supports both normal JPG and the custom SJPG formats.
- Decoding normal JPG consumes RAM with the size fo the whole uncompressed image (recommended only for devices with more RAM)
- SJPG is a custom format based on "normal" JPG and specially made for LVGL.
- SJPG is 'split-jpeg' which is a bundle of small jpeg fragments with an sjpg header.
- SJPG size will be almost comparable to the jpg file or might be a slightly larger.
- File read from file and c-array are implemented.
- SJPEG frame fragment cache enables fast fetching of lines if available in cache.
- By default the sjpg image cache will be image width * 2 * 16 bytes (can be modified)
- Currently only 16 bit image format is supported (TODO)
- Only the required portion of the JPG and SJPG images are decoded, therefore they can't be zoomed or rotated.
- 支持普通 JPG 和自定义 SJPG 格式。
- 解码普通 JPG 会消耗整个未压缩图像大小的 RAM (仅推荐用于具有更多 RAM 的设备)
- SJPG 是基于“普通” JPG 的自定义格式，专为 LVGL。
- SJPG 是“split-jpeg”，它是一捆带有 sjpg 标头的小 jpeg 片段。
- SJPG 大小几乎与 jpg 文件相当，或者可能稍大一些。
- 从文件和 c-array 读取的文件已实现。
- 如果缓存中可用，SJEPG 帧片段缓存可以快速获取行。
- 默认情况下，sjpg 图像缓存将为图像宽度 * 2 * 16 字节 (可以修改) - 目前仅支持 16 位图像格式 (TODO)
- 仅解码 JPG 和 SJPG 图像所需的部分，因此无法对其进行缩放或旋转。

8.3.2 Usage(用法)

If enabled in `lv_conf.h` by `LV_USE_SJPG` LVGL will register a new image decoder automatically so JPG and SJPG files can be directly used as image sources. For example:

如果在 `lv_conf.h` 中通过 `LV_USE_SJPG` 启用 LVGL 将自动注册一个新的图像解码器，因此 JPG 和 SJPG 文件可以直接用作图像源。例如：

```
lv_img_set_src(my_img, "S:/path/to/picture.jpg");
```

Note that, a file system driver needs to registered to open images from files. Read more about it [here](#) or just enable one in `lv_conf.h` with `LV_USE_FS_...`

请注意，文件系统驱动程序需要注册才能从文件中打开图像。阅读更多相关信息 [此处](#) 或仅在 `lv_conf.h` 中使用 `LV_USE_FS_...` 启用一个

8.3.3 Converter(转换器)

Converting JPG to C array(将 JPG 转换为 C 数组)

- Use lvgl online tool <https://lvgl.io/tools/imageconverter>
- Color format = RAW, output format = C Array
- 使用 lvgl 在线工具 <https://lvgl.io/tools/imageconverter>
- 颜色格式 = RAW，输出格式 = C 数组

Converting JPG to SJPG(将 JPG 转换为 SJPG)

python3 and the PIL library required. (PIL can be installed with `pip3 install pillow`)

To create SJPG from JPG:

- Copy the image to convert into `lvgl/scripts`
- `cd lvgl/scripts`
- `python3 jpg_to_sjpg.py image_to_convert.jpg`. It creates both a C files and an SJPG image.

The expected result is:

```
Conversion started...

Input:
    image_to_convert.jpg
    RES = 640 x 480

Output:
    Time taken = 1.66 sec
    bin size = 77.1 KB
    wallpaper.sjpg          (bin file)
    wallpaper.c              (c array)

All good!
```

8.3.4 Example(示例)

8.3.5 API

警告: doxygenfile: Cannot find file "lv_sjpg.h"

8.4 PNG decoder(PNG 解码器)

Allow the use of PNG images in LVGL. This implementation uses [lodepng](#) library.

If enabled in `lv_conf.h` by `LV_USE_PNG` LVGL will register a new image decoder automatically so PNG files can be directly used as any other image sources.

Note that, a file system driver needs to registered to open images from files. Read more about it [here](#) or just enable one in `lv_conf.h` with `LV_USE_FS_...`

The whole PNG image is decoded so during decoding RAM equals to `image width × image height × 4` bytes are required.

As it might take significant time to decode PNG images LVGL's [images caching](#) feature can be useful.

允许在 LVGL 中使用 PNG 图像。此实现使用 [lodepng](#) 库。

如果在 `lv_conf.h` 中通过 `LV_USE_PNG` 启用, LVGL 将自动注册一个新的图像解码器, 因此 PNG 文件可以直接用作任何其他图像源。

请注意, 需要注册文件系统驱动程序才能从文件中打开图像。阅读有关它的更多信息 [此处](#) 或仅使用 `LV_USE_FS_...` 在 `lv_conf.h` 中启用一个

整个 PNG 图像是因此在解码过程中解码 RAM 等于图像宽度 × 图像高度 × 4 字节是必需的。

因为解码 PNG 图像可能需要大量时间 LVGL 的 [图像缓存](<https://docs.lvgl.io/master/overview/image.html#image-caching>) 功能很有用。

8.4.1 Example(用法)

8.4.2 API

警告: doxygenfile: Cannot find file "lv_png.h"

8.5 GIF decoder(GIF 解码器)

Allow to use of GIF images in LVGL. Based on <https://github.com/lecrum/gifdec>

When enabled in `lv_conf.h` with `LV_USE_GIF` `lv_gif_create(parent)` can be used to create a gif widget. `lv_gif_set_src(obj, src)` works very similarly to `lv_img_set_src`. As source It also accepts images as variables (`lv_img_dsc_t`) or files.

允许在 LVGL 中使用 GIF 图像。基于 <https://github.com/lecrum/gifdec>

当在 `lv_conf.h` 和 `LV_USE_GIF` 中启用时, `lv_gif_create(parent)` 可用于创建 gif 小部件。

`lv_gif_set_src(obj, src)` 的工作方式与 `lv_img_set_src` 非常相似。作为源它还接受图像作为变量 (`lv_img_dsc_t`) 或文件。

8.5.1 Convert GIF files to C array(将 GIF 文件转换为 C 数组)

To convert a GIF file to byte values array use [LVGL's online converter](#). Select "Raw" color format and "C array" Output format.

要将 GIF 文件转换为字节值数组，请使用 [LVGL 的在线转换器](#)。选择“原始”颜色格式和“C 数组”输出格式。

8.5.2 Use GIF images from file(使用文件中的 GIF 图片)

For example:

```
lv_gif_set_src(obj, "S:/path/to/example.gif");
```

Note that, a file system driver needs to registered to open images from files. Read more about it [here](#) or just enable one in `lv_conf.h` with `LV_USE_FS_...`

请注意，文件系统驱动程序需要注册才能从文件中打开图像。阅读更多相关信息 [此处](#) 或仅在 `lv_conf.h` 中使用 `LV_USE_FS_...` 启用一个

8.5.3 Memory requirements(内存要求)

To decode and display a GIF animation the following amount of RAM is required:

- `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`: $5 \times \text{image width} \times \text{image height}$

要解码和显示 GIF 动画，需要以下 RAM 量：

- `LV_COLOR_DEPTH 8`: $3 \times \text{图像宽度} \times \text{图像高度}$
- `LV_COLOR_DEPTH 16`: $4 \times \text{图像宽度} \times \text{图像高度}$
- `LV_COLOR_DEPTH 32`: $5 \times \text{图像宽度} \times \text{图像高度}$

8.5.4 Example(用法)

8.5.5 API

```
警告: doxygenfile: Cannot find file "lv_gif.h"
```

8.6 FreeType support(支持 FreeType)

Interface to [FreeType](#) to generate font bitmaps run time.

[FreeType](#) 的接口以生成字体位图运行时。

8.6.1 Install FreeType(安装 FreeType)

- Download Freetype from [here](#)
- make
- sudo make install
- 从[这里](#)下载 Freetype
- make
- sudo make install

8.6.2 Add FreeType to your project(将 FreeType 添加到项目中)

- Add include path: /usr/include/freetype2 (for GCC: -I/usr/include/freetype2 -L/usr/local/lib)
- Add library: freetype (for GCC: -L/usr/local/lib -lfreetype)
- 添加包含路径: /usr/include/freetype2 (对于 GCC: -I/usr/include/freetype2 -L/usr/local/lib)
- 添加库: freetype (对于 GCC: -L /usr/local/lib -lfreetype)

8.6.3 Usage(用法)

Enable `LV_USE_FREETYPE` in `lv_conf.h`.

See the examples below.

Note that, 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.

在 `lv_conf.h` 中启用 `LV_USE_FREETYPE`。

参见下面的示例。

注意, FreeType 扩展不使用 LVGL 的文件系统。您可以像往常一样在操作系统或平台上简单地传递字体的路径。

8.6.4 Learn more(深入学习)

- FreeType [tutorial](#)
- LVGL's font interface
- FreeType [教程](#)
- LVGL 的 [字体界面](<https://docs.lvgl.io/v7/en/html/overview/font.html#add-a-new-font-engine>)

8.6.5 API

警告: doxygenfile: Cannot find file "lv_freetype.h"

8.7 QR code(二维码)

QR code generation with LVGL. Uses QR-Code-generator by nayuki.

使用 LVGL 生成二维码。使用 QR-Code-generator by nayuki。

8.7.1 Get started(开始)

- Download or clone this repository
 - [Download](#) from GitHub
 - Clone: git clone https://github.com/lvgl/lv_lib_qrcode.git
- Include the library: `#include "lv_lib_qrcode/lv_qrcode.h"`
- Test with the following code:
- 下载或克隆此存储库
 - [下载](#) 从 GitHub
 - 克隆: git clone https://github.com/lvgl/lv_lib_qrcode.git
- 引用库: `#include "lv_lib_qrcode/lv_qrcode.h"`
- 使用以下代码进行测试:

```
const char * data = "Hello world";

/*Create a 100x100 QR code*/
lv_obj_t * qr = lv_qrcode_create(lv_scr_act(), 100, lv_color_hex3(0x33f), lv_color_
hex3(0xeeff));

/*Set data*/
lv_qrcode_update(qr, data, strlen(data));
```

8.7.2 Notes(注意)

- QR codes with less data are smaller but they scaled by an integer numbers number to best fit to the given size
- 数据较少的二维码较小，但按整数缩放以最适合给定大小

8.7.3 Example(示例)

8.7.4 API

警告: doxygenfile: Cannot find file "lv_qrcode.h"

8.8 Lottie player

Allows to use Lottie animations in LVGL. Taken from this [base repository](#)

LVGL provides the interface to [Samsung/rllottie](#) library's C API. That is the actual Lottie player is not part of LVGL, it needs to be built separately.

允许在 LVGL 中使用 Lottie 动画。取自 [base repository](#)

LVGL 提供了到 [Samsung/rllottie](#) 库的 C API 的接口。也就是说实际的 Lottie 播放器不是 LVGL 的一部分，它需要单独构建。

8.8.1 Build Rlottie(构建 Rlottie)

To build Samsung's Rlottie C++14-compatible compiler and optionally CMake 3.14 or higher is required.

To build on desktop you can follow the instructions from Rlottie's [README](#). In the most basic case it looks like this:

要构建三星的 Rlottie C++14 兼容编译器和可选的 CMake 3.14 或更高版本是必需的。

要在桌面上构建，您可以按照 Rlottie 的 [\[README\]\(https://github.com/Samsung/rllottie/blob/master/README.md\)](#)。在最基本的情况下，它看起来像这样：

```
mkdir rlottie_workdir
cd rlottie_workdir
git clone https://github.com/Samsung/rllottie.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.

最后将 `-lrlottie` 标志添加到您的链接器。

在嵌入式系统上，您需要注意将 Rlottie 集成到给定的构建系统。

8.8.2 Usage(用法)

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`.

The `width` and `height` of the object be set in the `create` function and the animation will be scaled accordingly.

您可以使用来自文件或原始数据（文本）的动画。在任何一种情况下，

您首先需要在 `lv_conf.h` 中启用 `LV_USE_RLOTTIE`。对象的 `width` 和 `height` 在 `create` 函数中设置，动画将相应缩放。

Use Rlottie from file(使用文件中的 Rlottie)

To create a Lottie animation from file use:

要从文件创建 Lottie 动画，请使用：

```
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.

请注意，Rlottie 使用标准 STUDIO C 文件 API，因此您可以“正常”使用路径，并且不需要特定于 LVGL 的驱动程序字母。

Use Rlottie from raw string data(使用原始字符串数据中的 Rlottie)

`lv_example_rlottie_approve.c` contains an example animation in raw format. Instead storing the JSON string a hex array is stored for the following reasons:

- avoid escaping " 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. E.g.:

`lv_example_rlottie_approve.c` 包含原始格式的示例动画。代替存储 JSON 字符串，存储十六进制数组的原因如下：

- 避免在 JSON 文件中转义 "
- 一些编译器不支持很长的字符串

`lvgl/scripts/filetohex.py` 可用于转换一个 Lottie 归档一个十六进制数组。例如：

```
./filetohex.py path/to/lottie.json > out.txt
```

To create an animation from raw data:

从原始数据创建一个动画：

```
extern const uint8_t lottie_data[];
lv_obj_t* lottie = lv_rlottie_create_from_raw(parent, width, height, (const char*)
↳ *)lottie_data);
```

8.8.3 Getting animations(获取动画)

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.

Lottie 是标准和流行的格式，因此您可以在网络上找到许多动画文件。

例如：<https://lottiefiles.com/> 您还可以使用 Adobe After Effects 或类似软件创建自己的动画。

8.8.4 Example(示例)

8.8.5 API

警告： doxygenfile: Cannot find file "lv_rlottie.h"

Others (其他)

9.1 Snapshot (快照)

Snapshot provides APIs to take snapshot image for LVGL object together with its children. The image will look exactly like the object.

快照为 LVGL 对象及其子实体提供了获取快照图像的 API。该图像将看起来与该对象完全一样。

9.1.1 Usage (用法)

Simply call API `lv_snapshot_take` to generate the image descriptor which can be set as image object src using `lv_img_set_src`.

Note, only below color formats are supported for now:

- `LV_IMG_CF_TRUE_COLOR_ALPHA`
- `LV_IMG_CF_ALPHA_1BIT`
- `LV_IMG_CF_ALPHA_2BIT`
- `LV_IMG_CF_ALPHA_4BIT`
- `LV_IMG_CF_ALPHA_8BIT`

只需调用 API `lv_snapshot_take` 即可生成图像描述符，使用 `lv_img_set_src` 可将其设置为图像对象 src。

注意，目前只支持以下颜色格式。

- `LV_IMG_CF_TRUE_COLOR_ALPHA`
- `LV_IMG_CF_ALPHA_1BIT`
- `LV_IMG_CF_ALPHA_2BIT`
- `LV_IMG_CF_ALPHA_4BIT`

- LV_IMG_CF_ALPHA_8BIT

Free the Image (释放图像)

The memory `lv_snapshot_take` uses are dynamically allocated using `lv_mem_alloc`. Use API `lv_snapshot_free` to free the memory it takes. This will firstly free memory the image data takes, then the image descriptor.

Take caution to free the snapshot but not delete the image object. Before free the memory, be sure to firstly unlink it from image object, using `lv_img_set_src(NULL)` and `lv_img_cache_invalidate_src(src)`.

Below code snippet explains usage of this API.

`lv_snapshot_take` 使用的内存是用 `lv_mem_alloc` 动态分配的。 使用 API `lv_snapshot_free` 来释放它所占用的内存。 这将首先释放图像数据占用的内存，然后是图像描述符。

请注意释放快照，但不要删除图像对象。 在释放内存之前，一定要先解除它与图像对象的链接，使用 `lv_img_set_src(NULL)` 和 `lv_img_cache_invalidate_src(src)`。

下面的代码片段解释了这个 API 的用法。

```
void update_snapshot(lv_obj_t * obj, lv_obj_t * img_snapshot)
{
    lv_img_dsc_t* snapshot = (void*)lv_img_get_src(img_snapshot);
    if(snapshot) {
        lv_snapshot_free(snapshot);
    }
    snapshot = lv_snapshot_take(obj, LV_IMG_CF_TRUE_COLOR_ALPHA);
    lv_img_set_src(img_snapshot, snapshot);
}
```

Use Existing Buffer (使用现有缓冲区)

If the snapshot needs update now and then, or simply caller provides memory, use API `lv_res_t lv_snapshot_take_to_buf(lv_obj_t * obj, lv_img_cf_t cf, lv_img_dsc_t * dsc, void * buf, uint32_t buff_size)`; for this case. It's caller's responsibility to alloc/free the memory.

If snapshot is generated successfully, the image descriptor is updated and image data will be stored to provided `buf`.

Note that snapshot may fail if provided buffer is not enough, which may happen when object size changes. It's recommended to use API `lv_snapshot_buf_size_needed` to check the needed buffer size in byte firstly and resize the buffer accordingly.

如果快照偶尔需要更新，或者仅仅是调用者提供内存，在这种情况下可以使用 API `lv_res_t lv_snapshot_take_to_buf(lv_obj_t * obj, lv_img_cf_t cf, lv_img_dsc_t * dsc, void * buf, uint32_t buff_size)`。 分配/释放内存是调用者的责任。

如果快照成功生成，图像描述符将被更新，图像数据将被存储到提供的 `buf`。

注意，如果提供的缓冲区不够，快照可能会失败，这可能发生在对象大小改变时。 建议使用 API `lv_snapshot_buf_size_needed` 首先检查所需的缓冲区的字节数，并相应地调整缓冲区的大小。

9.1.2 Example

9.1.3 API

警告: doxygenfile: Cannot find file "lv_snapshot.h"

CHAPTER 10

Contributing (贡献)

10.1 Introduction (介绍)

Join LVGL's community and leave your footprint in the library!

There are a lot of ways to contribute to LVGL even if you are new to the library or even new to programming.

It might be scary to make the first step but you have nothing to be afraid of. A friendly and helpful community is waiting for you. Get to know like-minded people and make something great together.

So let's find which contribution option fits you the best and help you join the development of LVGL!

Before getting started here are some guidelines to make contribution smoother:

- Be kind and friendly.
- Be sure to read the relevant part of the documentation before posting a question.
- Ask questions in the [Forum](#) and use [GitHub](#) for development-related discussions.
- Always fill out the post or issue templates in the Forum or GitHub (or at least provide equivalent information). It makes understanding your contribution or issue easier and you will get a useful response faster.
- If possible send an absolute minimal but buildable code example in order to reproduce the issue. Be sure it contains all the required variable declarations, constants, and assets (images, fonts).
- Use [Markdown](#) to format your posts. You can learn it in 10 minutes.
- Speak about one thing in one issue or topic. It makes your post easier to find later for someone with the same question.
- Give feedback and close the issue or mark the topic as solved if your question is answered.
- For non-trivial fixes and features, it's better to open an issue first to discuss the details instead of sending a pull request directly.
- Please read and follow the Coding style guide.

加入 LVGL 的社区，在图书馆留下你的足迹！

有很多方法可以为 LVGL 做出贡献，即使您是库的新手，甚至是编程的新手。

迈出第一步可能会很可怕，但你没有什么可害怕的。一个友好而乐于助人的社区正等着您。结识志同道合的人，共同创造美好。

那么让我们来找出最适合您的贡献选项，并帮助您加入 LVGL 的开发！

在开始之前，这里有一些指导方针可以使贡献更顺畅：

- 善良和友好。
- 在发布问题之前，请务必阅读文档的相关部分。
- 在[论坛](#)提出问题，并使用[GitHub](#)进行与开发相关的讨论。
- 始终填写论坛或 GitHub 中的帖子或问题模板（或至少提供等效信息）。它使您更容易理解您的贡献或问题，并且您将更快地获得有用的响应。
- 如果可能，发送一个绝对最小但可构建的代码示例以重现问题。确保它包含所有必需的变量声明、常量和资产（图像、字体）。
- 使用[Markdown](#)来格式化您的帖子。你可以在 10 分钟内学会它。
- 在一个问题或主题中谈论一件事。以后有相同问题的人可以更轻松地找到您的帖子。
- 如果您的问题得到解答，请提供反馈并关闭问题或将主题标记为已解决。
- 对于重要的修复和功能，最好先打开一个问题来讨论细节，而不是直接发送拉取请求。
- 请阅读并遵循编码风格指南。

10.2 Pull request（拉取请求）

Merging new code into the lvgl, documentation, blog, examples, and other repositories happen via *Pull requests* (PR for short). A PR is a notification like "Hey, I made some updates to your project. Here are the changes, you can add them if you want." To do this you need a copy (called fork) of the original project under your account, make some changes there, and notify the original repository about your updates. You can see what it looks like on GitHub for LVGL here: <https://github.com/lvgl/lvgl/pulls>.

To add your changes you can edit files online on GitHub and send a new Pull request from there (recommended for small changes) or add the updates in your favorite editor/IDE and use git to publish the changes (recommended for more complex updates).

将新代码合并到 lvgl、文档、博客、示例和其他存储库中是通过 *Pull 请求 *（简称 PR）进行的。PR 是类似于“嘿，我对您的项目进行了一些更新。这是更改，如果需要，您可以添加它们”的通知。为此，您需要您帐户下的原始项目的副本（称为 fork），在那里进行一些更改，并将您的更新通知原始存储库。您可以在 GitHub 上查看 LVGL 的样子：<https://github.com/lvgl/lvgl/pulls>。

要添加您的更改，您可以在 GitHub 上在线编辑文件并从那里发送新的拉取请求（建议进行小改动）或在您喜欢的编辑器/IDE 中添加更新并使用 git 发布更改（推荐用于更复杂的更新）。

10.2.1 From GitHub (来自 GitHub)

1. Navigate to the file you want to edit.
 2. Click the Edit button in the top right-hand corner.
 3. Add your changes to the file.
 4. Add a commit message on the bottom of the page.
 5. Click the *Propose changes* button.
1. 导航到要编辑的文件。
 2. 单击右上角的编辑按钮。
 3. 将您的更改添加到文件中。
 4. 在页面底部添加提交信息。
 5. 单击提议更改按钮。

10.2.2 From command line (从命令行获取)

The instructions describe the main `lvgl` repository but it works the same way for the other repositories.

1. Fork the `lvgl` repository. To do this click the "Fork" button in the top right corner. It will "copy" the `lvgl` repository to your GitHub account (https://github.com/<YOUR_NAME>?tab=repositories)
2. Clone your forked repository.
3. Add your changes. You can create a *feature branch* from *master* for the updates: `git checkout -b the-new-feature`
4. Commit and push your changes to the forked `lvgl` repository.
5. Create a PR on GitHub from the page of your `lvgl` repository (https://github.com/<YOUR_NAME>/lvgl) by clicking the "*New pull request*" button. Don't forget to select the branch where you added your changes.
6. Set the base branch. It means where you want to merge your update. In the `lvgl` repo fixes go to `master`, new features to `dev` branch.
7. Describe what is in the update. An example code is welcome if applicable.
8. If you need to make more changes, just update your forked `lvgl` repo with new commits. They will automatically appear in the PR.

这些说明描述了主要的 `lvgl` 存储库，但它的工作方式与其他存储库相同。

1. fork `lvgl` 仓库。为此，请单击右上角的“叉”按钮。它会将 `lvgl` 存储库“复制”到您的 GitHub 帐户 (https://github.com/<YOUR_NAME>?tab=repositories)
2. 克隆您的分叉存储库。
3. 添加您的更改。您可以从 `master` 创建一个 *feature* 分支用于更新: `git checkout -b the-new-feature`
4. 提交并将您的更改推送到分叉的 `lvgl` 存储库。
5. 在你的 `lvgl` 存储库页面 (https://github.com/<YOUR_NAME>/lvgl) 上点击 "*New pull request*" 按钮，在 GitHub 上创建 PR。不要忘记选择添加更改的分支。
6. 设置基础分支。这意味着您要合并更新的位置。在 `lvgl` 存储库中，修复转到 `master`, `dev` 分支的新功能。

7. 描述更新内容。如果适用，欢迎提供示例代码。
8. 如果您需要进行更多更改，只需使用新提交更新您的分叉 `lvgl` 存储库。它们将自动出现在 PR 中。

10.2.3 Commit message format (commit 的格式)

In commit messages please follow the [Angular Commit Format](#).

Some examples:

在提交消息中，请遵循 [Angular Commit Format](#)。

一些例子：

```
fix(img) update size if 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

```
docs(porting) fix typo
```

10.3 Developer Certification of Origin (DCO)(开发者原产地认证 (DCO))

10.3.1 Overview (概述)

To ensure all licensing criteria are met for every repository of the LVGL project, we apply a process called DCO (Developer's Certificate of Origin).

The text of DCO can be read here: <https://developercertificate.org/>.

By contributing to any repositories of the LVGL project you agree that your contribution complies with the DCO.

If your contribution fulfills the requirements of the DCO no further action is needed. If you are unsure feel free to ask us in a comment.

为确保 LVGL 项目的每个存储库都满足所有许可标准，我们应用了一个称为 DCO（开发者原产地证书）的流程。

DCO 的文本可以在这里阅读：<https://developercertificate.org/>。

通过为 LVGL 项目的任何存储库做出贡献，您同意您的贡献符合 DCO。

如果您的捐款符合 DCO 的要求，则无需采取进一步行动。如果您不确定，请随时在评论中询问我们。

10.3.2 Accepted licenses and copyright notices (接受的许可和版权声明)

To make the DCO easier to digest, here are some practical guides about specific cases:

为了让 DCO 更容易消化，这里有一些关于特定案例的实用指南：

Your own work (你自己的作品)

The simplest case is when the contribution is solely your own work. In this case you can just send a Pull Request without worrying about any licensing issues.

最简单的情况是贡献完全是您自己的工作。在这种情况下，您可以只发送拉取请求而不必担心任何许可问题。

Use code from online source (使用来自网上的代码)

If the code you would like to add is based on an article, post or comment on a website (e.g. StackOverflow) the license and/or rules of that site should be followed.

For example in case of StackOverflow a notice like this can be used:

如果您要添加的代码基于网站（例如 StackOverflow）上的文章、帖子或评论，则应遵循该网站的许可和/或规则。

例如，在 StackOverflow 的情况下，可以使用这样的通知：

```
/* 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 ...
```

Use MIT licensed code (使用 MIT 许可代码)

As LVGL is MIT licensed, other MIT licensed code can be integrated without issues. The MIT license requires a copyright notice be added to the derived work. Any derivative work based on MIT licensed code must copy the original work's license file or text.

由于 LVGL 是 MIT 许可的，因此可以毫无问题地集成其他 MIT 许可代码。MIT 许可证要求在衍生作品中添加版权声明。任何基于 MIT 许可代码的衍生作品必须复制原始作品的许可文件或文本。

Use GPL licensed code (使用 GPL 许可代码)

The GPL license is not compatible with the MIT license. Therefore, LVGL can not accept GPL licensed code.

GPL 许可证与 MIT 许可证不兼容。因此，LVGL 不能接受 GPL 许可代码。

10.4 Ways to contribute (贡献方式)

Even if you're just getting started with LVGL there are plenty of ways to get your feet wet. Most of these options don't even require knowing a single line of LVGL code.

Below we have collected some opportunities about the ways you can contribute to LVGL.

即使您刚刚开始使用 LVGL，也有很多方法可以让您的脚变湿。大多数这些选项甚至不需要知道一行 LVGL 代码。

下面我们收集了一些关于您可以为 LVGL 做出贡献的方式的机会。

10.4.1 Give LVGL a Star (给 LVGL 一颗星)

Show that you like LVGL by giving it star on GitHub!

Star

This simple click makes LVGL more visible on GitHub and makes it more attractive to other people. So with this, you already helped a lot!

通过在 GitHub 上给它 Star 来表明你喜欢 LVGL！

```
<a class="github-button" href="https://github.com/lvgl/lvgl" data-icon="octicon-star" data-size="large" data-show-count="true" data-label="Star lvgl/lvgl on GitHub">Star
```

这个简单的点击使 LVGL 在 GitHub 上更显眼，并使其对其他人更具吸引力。所以有了这个，你已经帮了很多忙！

10.4.2 Tell what you have achieved (讲述你的成就)

Have you already started using LVGL in a *Simulator*, a development board, or on your custom hardware? Was it easy or were there some obstacles? Are you happy with the result? Showing your project to others is a win-win situation because it increases your and LVGL's reputation at the same time.

You can post about your project on Twitter, Facebook, LinkedIn, create a YouTube video, and so on. Only one thing: On social media don't forget to add a link to <https://lvgl.io> or <https://github.com/lvgl> and use the hashtag #lvgl. Thank you! :)

You can also open a new topic in the [My projects](#) category of the Forum.

The [LVGL Blog](#) welcomes posts from anyone. It's a good place to talk about a project you created with LVGL, write a tutorial, or share some nice tricks. The latest blog posts are shown on the [homepage of LVGL](#) to make your work more visible.

The blog is hosted on GitHub. If you add a post GitHub automatically turns it into a website. See the [README](#) of the blog repo to see how to add your post.

Any of these help to spread the word and familiarize new developers with LVGL.

If you don't want to speak about your project publicly, feel free to use [Contact form](#) on lvgl.io to private message to us.

您是否已经开始在[模拟器](#)、[开发板](#)或[自定义硬件](#)中使用 LVGL？是容易还是有什么障碍？您对结果满意吗？向他人展示您的项目是一个双赢的局面，因为它同时增加了您和 LVGL 的声誉。

您可以在 Twitter、Facebook、LinkedIn 上发布您的项目，创建 YouTube 视频等。只有一件事：在社交媒体上不要忘记添加一个链接到 <https://lvgl.io> 或 <https://github.com/lvgl> 并使用主题标签 #lvgl。谢谢！:)

您也可以在论坛的[我的项目](#)类别中打开一个新主题。

LVGL 博客 欢迎任何人发帖。这是谈论您使用 LVGL 创建的项目、编写教程或分享一些不错的技巧的好地方。最新博文展示在【LVGL 首页】(<https://lvgl.io>)，让你的作品更显眼。

该博客托管在 GitHub 上。如果您添加帖子，GitHub 会自动将其转换为网站。请参阅博客存储库的 README 以了解如何添加您的帖子。

其中任何一项都有助于传播信息并使新开发人员熟悉 LVGL。

如果您不想公开谈论您的项目，请随时使用 lvgl.io 上的联系表格 私信给我们。

10.4.3 Write examples (撰写实例)

As you learn LVGL you will probably play with the features of widgets. Why not publish your experiments?

Each widgets' documentation contains examples. For instance, here are the examples of the Drop-down list widget. The examples are directly loaded from the `lvgl/examples` folder.

So all you need to do is send a *Pull request* to the `lvgl` repository and follow some conventions:

- Name the examples like `lv_example_<widget_name>_<index>`.
- Make the example as short and simple as possible.
- Add comments to explain what the example does.
- Use 320x240 resolution.
- Update `index.rst` in the example's folder with your new example. To see how other examples are added, look in the `lvgl/examples/widgets` folder.

当您学习 LVGL 时，您可能会使用小部件的功能。为什么不发表你的实验？

每个小部件的文档都包含示例。例如，这里是 下拉列表 小部件的示例。示例直接从 `lvgl/examples` 文件夹加载。

所以你需要做的就是向 `lvgl` 存储库发送一个 *Pull request* 并遵循一些约定：

- 将示例命名为 `lv_example_<widget_name>_<index>`。
- 使示例尽可能简短。
- 添加注释以解释示例的作用。
- 使用 320x240 分辨率。
- 使用您的新示例更新示例文件夹中的 `index.rst`。要查看其他示例是如何添加的，请查看 `lvgl/examples/widgets` 文件夹。

10.4.4 Improve the docs (改进文档)

As you read the documentation you might see some typos or unclear sentences. All the documentation is located in the `lvgl/docs` folder. For typos and straightforward fixes, you can simply edit the file on GitHub.

Note that the documentation is also formatted in `Markdown`.

当您阅读文档时，您可能会看到一些拼写错误或不清楚的句子。所有文档都位于 `lvgl/docs` 文件夹中。对于拼写错误和直接修复，您只需在 GitHub 上编辑文件即可。

请注意，文档的格式也为 `Markdown`。

10.4.5 Report bugs (报告 bug)

As you use LVGL you might find bugs. Before reporting them be sure to check the relevant parts of the documentation.

If it really seems like a bug feel free to open an issue on [GitHub](#).

When filing the issue be sure to fill out the template. It helps find the root of the problem while avoiding extensive questions and exchanges with other developers.

当您使用 LVGL 时，您可能会发现错误。在报告它们之前，请务必检查文档的相关部分。

如果它真的看起来像是一个错误，请随时打开 [GitHub 上的问题](#)。

提交问题时，请务必填写模板。它有助于找到问题的根源，同时避免广泛的问题和与其他开发人员的交流。

10.4.6 Send fixes (提交补丁)

The beauty of open-source software is you can easily dig in to it to understand how it works. You can also fix or adjust it as you wish.

If you found and fixed a bug don't hesitate to send a [Pull request](#) with the fix.

In your Pull request please also add a line to [CHANGELOG.md](#).

开源软件的美妙之处在于您可以轻松深入了解它的工作原理。您也可以根据需要修复或调整它。

如果您发现并修复了错误，请毫不犹豫地发送带有修复程序的 [Pull request](#)。

在您的拉取请求中，还请在 [CHANGELOG.md](#) 中添加一行。

10.4.7 Join the conversations in the Forum (参与论坛讨论)

It feels great to know you are not alone if something is not working. It's even better to help others when they struggle with something.

While you were learning LVGL you might have had questions and used the Forum to get answers. As a result, you probably have more knowledge about how LVGL works.

One of the best ways to give back is to use the Forum and answer the questions of newcomers - like you were once.

Just read the titles and if you are familiar with the topic don't hesitate to share your thoughts and suggestions.

Participating in the discussions is one of the best ways to become part of the project and get to know like-minded people!

如果某些事情不起作用，知道您并不孤单，这感觉很棒。在别人遇到困难时帮助他们甚至更好。

当您学习 LVGL 时，您可能会遇到问题并使用论坛来获得答案。因此，您可能对 LVGL 的工作原理有了更多的了解。

回馈的最佳方式之一是使用论坛并回答新人的问题 - 就像您曾经一样。

只需阅读标题，如果您熟悉该主题，请随时分享您的想法和建议。

参与讨论是成为项目的一部分并结识志同道合的人的最佳方式之一！

10.4.8 Add features (添加功能)

If you have created a cool widget, or added useful feature to LVGL feel free to open a new PR for it. We collect the optional features (a.k.a. plugins) in `lvgl/src/extra` folder so if you are interested in adding a new features please use this folder. The `README` file describes the basics rules of contribution and also lists some ideas.

For further ideas take a look at the our [Roadmap](#) page. If you are interested in any of them feel free to share your opinion and/or participate in the implementation.

Other features which are (still) not on the road map are listed in the `Feature request` category of the Forum.

When adding a new features the followings also needs to be updated:

- Update `lv_conf_template.h`
- Add description in the `docs`
- Add `examples`
- Update the `changelog`

如果您创建了一个很酷的小部件，或者为 LVGL 添加了有用的功能，请随时为它打开一个新的 PR。我们在 `lvgl/src/extra` 文件夹中收集可选功能（又名插件），因此如果您有兴趣添加新功能，请使用这个文件夹。`README` 文件描述了贡献的基本规则，并列出了一些想法。

有关更多想法，请查看我们的[Roadmap](#) 页面。如果您对其中任何一个感兴趣，请随时分享您的意见和/或参与实施。

其他（仍然）不在路线图中的功能列在论坛的 `功能请求` 类别中。

添加新功能时，还需要更新以下内容：

- 更新 `lv_conf_template.h`
- 在 `docs` 中添加说明
- 添加示例
- 更新 `变更日志`

10.4.9 Become a maintainer (成为维护者)

If you want to become part of the core development team, you can become a maintainer of a repository.

By becoming a maintainer:

- You get write access to that repo:
 - Add code directly without sending a pull request
 - Accept pull requests
 - Close/reopen/edit issues
- Your input has higher impact when we are making decisions

You can become a maintainer by invitation, however the following conditions need to met

1. Have > 50 replies in the Forum. You can look at your stats [here](#)
2. Send > 5 non-trivial pull requests to the repo where you would like to be a maintainer

If you are interested, just send a message (e.g. from the Forum) to the current maintainers of the repository. They will check if the prerequisites are met. Note that meeting the prerequisites is not a guarantee of acceptance, i.e. if the conditions are met you won't automatically become a maintainer. It's up to the current maintainers to make the decision.

如果您想成为核心开发团队的一员，您可以成为存储库的维护者。

通过成为维护者：

- 您可以对该存储库进行写访问：
 - 无需发送拉取请求，直接添加代码
 - 接受拉取请求
 - 关闭/重新打开/编辑问题
- 当我们做出决定时，您的意见会产生更大的影响

您可以通过邀请成为维护者，但需要满足以下条件

1. 论坛回复超过 50 条。你可以查看你的统计数据[这里](#)
2. 向您希望成为维护者的存储库发送 > 5 个重要的拉取请求

如果您有兴趣，只需向存储库的当前维护者发送一条消息（例如来自论坛）。他们将检查是否满足先决条件。请注意，满足先决条件并不能保证被接受，即如果满足条件，您将不会自动成为维护者。由当前的维护者做出决定。

10.4.10 Move your project repository under LVGL organization (将您的项目库移到 LVGL 组织下)

Besides the core `lvgl` repository there are other repos for ports to development boards, IDEs or other environment. If you ported LVGL to a new platform we can host it under the LVGL organization among the other repos.

This way your project will become part of the whole LVGL project and can get more visibility. If you are interested in this opportunity just open an issue in [lvgl repo](#) and tell what you have!

If we agree that your port fit well into the LVGL organization, we will open a repository for your project where you will have admin rights.

To make this concept sustainable there a few rules to follow:

- You need to add a README to your repo.
- We expect to maintain the repo to some extent:
 - Follow at least the major versions of LVGL
 - Respond to the issues (in a reasonable time)
- If there is no activity in a repo for 1 year it will be archived

除了核心 `lvgl` 存储库，还有其他存储库用于开发板、IDE 或其他环境的端口。如果您将 LVGL 移植到一个新平台，我们可以将它托管在 LVGL 组织下的其他存储库中。

这样您的项目将成为整个 LVGL 项目的一部分，并可以获得更多的可见性。如果您对这个机会感兴趣，只需打开一个 [问题在 lvgl repo](#) 并告诉您有什么！

如果我们同意您的移植非常适合 LVGL 组织，我们将为您的项目打开一个存储库，您将在其中拥有管理员权限。

为了使这个概念可持续，需要遵循一些规则：

- 你需要在你的 repo 中添加一个 README。
- 我们希望在一定程度上维持回购：
 - 至少遵循 LVGL 的主要版本
 - 回应问题（在合理的时间内）

- 如果 1 年内没有任何活动，它将被存档

Changelog (更新日志)

11.1 v8.1.0 (In progress) (进行中)

- feat(img) add img_size property (#2284) fe461caf
- feat(calendar) improve MicroPython example (#2366) Amir Gonnen 5f6e07e5
- feat(obj) add lv_obj_del_delayed() c6a2e15e
- feat(event, widgets) improve the paramter of LV_EVENT_DRAW_PART_BEGIN/END Gabor Kiss-Vamosi 88c48594
- feat(led) send LV_EVENT_DRAW_PART_BEGIN/END Gabor Kiss-Vamosi fcd4aa39
- feat(obj) send LV_EVENT_DRAW_PART_BEGIN/END for MAIN and SCROLLBAR parts Gabor Kiss-Vamosi b203167c
- feat(spinbox) add function to set cursor to specific position (#2314) dyktronix 7066c8fb
- feat(timer) check if lv_tick_inc is called aa6641a6
- feat(docs) add view on GitHub link a716ac6e
- feat(event) pass the scroll aniamtion to LV_EVENT_SCROLL_BEGIN ca54ecfe
- feat(img) 添加 img_size 属性 (#2284) fe461caf
- feat(calendar) 改进 MicroPython 示例 (#2366) Amir Gonnen 5f6e07e5
- feat(obj) 添加 lv_obj_del_delayed() c6a2e15e
- feat(event, widgets) 改进了 LV_EVENT_DRAW_PART_BEGIN/END Gabor Kiss-Vamosi 88c48594 的参数
- feat(led) 发送 LV_EVENT_DRAW_PART_BEGIN/END Gabor Kiss-Vamosi fcd4aa39
- feat(obj) 发送 LV_EVENT_DRAW_PART_BEGIN/END for MAIN 和 SCROLLBAR 部分 Gabor Kiss-Vamosi b203167c
- feat(spinbox) 添加将光标设置到特定位置的功能 (#2314) dyktronix 7066c8fb
- feat(timer) 检查 lv_tick_inc 是否被调用 aa6641a6

- feat(docs) 在 GitHub 链接上添加视图 a716ac6e
- feat(event) 将滚动动画传递给 LV_EVENT_SCROLL_BEGIN ca54ecfe

11.2 v8.0.2 (16.07.2021)

- fix(theme) improve button focus of keyboard
- fix(tabview) send LV_EVENT_VALUE_CHANGED only once
- fix(imgbtn) use the correct src in LV_EVENT_GET_SELF_SIZE
- fix(color) remove extraneous cast for 8-bit color
- fix(obj style) fix children reposition if the parent's padding changes.
- fix(color) remove extraneous _LV_COLOR_MAKE_TYPE_HELPER (#2372)
- fix(spinner) should not be clickable (#2373)
- fix(obj) improve how the focusing indev is determined
- fix(template) update indev template for v8
- fix(sprintf) skip defining attribute if pycparser is used
- refactor(sprintf) add printf-like function attribute to _lv_txt_set_text_vfmt and lv_label_set_text_fmt (#2332)
- fix(template) include lvgl.h in lv_port_*_template.c files
- fix(obj) detecting which indev sent LV_EVENT_FOCUS
- fix(span) fill LV_EVENT_GET_SELF_SIZE (#2360)
- fix(arc) disable LV_OBJ_FLAG_SCROLL_CHAIN by default
- fix(draw) fix arc bg image drawing with full arcs
- fix(disp) fix memory leak in lv_disp_remove (#2355)
- fix warnings introduced by 3fb8baf5
- fix(widgets) use lv_obj_class for all the widgets
- fix(obj) move clean ups from lv_obj_del to lv_obj_destructor
- fix(roller) fix partial redraw of the selected area
- fix(roller) adjust the size of the selected area correctly
- fix(obj) delete useless type conversion (#2343)
- fix(lv_obj_scroll.h) typos (#2345)
- fix(scroll) fire LV_EVENT_SCROLL_BEGIN in the same spot for both axes
- fix(btnmatrix) fix button invalidation on focus change
- fix(textarea) style update in oneline mode + improve scroll to cursor
- fix(tlsf) do not use <assert.h>
- fix(imgbtn) consider width==LV_SIZE_CONTENT if only mid. img is set
- fix(refr) reduce the nesting level in lv_refr_area
- fix(txt) enhance the function of break_chars (#2327)

- fix(pxp): update RTOS macro for SDK 2.10
- fix(vglite): update for v8
- fix(pxp): update for v8
- fix(flex) fix layout update and invalidation issues
- fix(flex) fix NULL pointer dereference
- fix(obj, switch) do not send LV_EVENT_VALUE_CHANGED twice
- fix(color) overflow with 16 bit color depth
- fix(coords) fix using large coordinates
- fix(chart) fix crash if no series are added
- fix(chart) invalidation with LV_CHART_UPDATE_MODE_SHIFT
- fix(align) fix lv_obj_align_to G
- fix(table) invalidate the table on cell value change
- fix(label) remove duplicated lv_obj_refresh_self_size
- fix(draw) underflow in subpixel font drawing
- fix(scroll) do not send unnecessary scroll end events
- 修复（主题）改善键盘的按钮焦点
- 修复（tabview）只发送一次 LV_EVENT_VALUE_CHANGED
- fix(imgbtn) 在 LV_EVENT_GET_SELF_SIZE 中使用正确的 src
- fix(color) 移除 8 位颜色的无关投射
- fix(obj style) 如果父级的填充发生变化，则修复子级重新定位。
- 修复（颜色）删除无关的 _LV_COLOR_MAKE_TYPE_HELPER (#2372)
- 修复（微调器）不应该是可点击的 (#2373)
- fix(obj) 改进如何确定聚焦 indev
- 修复（模板）更新 v8 的 indev 模板
- 修复（printf）如果使用 pycparser 则跳过定义属性
- 重构（printf）将类似 printf 的函数属性添加到 _lv_txt_set_text_vfmt 和 lv_label_set_text_fmt (#2332)
- 修复（模板）在 lv_port_*_template.c 文件中包含 lvgl.h
- fix(obj) 检测哪个 indev 发送了 LV_EVENT_FOCUS
- 修复（跨度）填充 LV_EVENT_GET_SELF_SIZE (#2360)
- 修复（弧）默认禁用 LV_OBJ_FLAG_SCROLL_CHAIN
- 修复（绘制）修复带有完整弧线的弧形背景图像绘制
- fix(disp) 修复 lv_disp_remove 中的内存泄漏 (#2355)
- 修复 3fb8baf5 引入的警告
- 修复（小部件）对所有小部件使用 lv_obj_class
- fix(obj) 将清理从 lv_obj_del 移动到 lv_obj_destructor
- fix(roller) 修复选定区域的部分重绘

- 修复（滚轮）正确调整所选区域的大小
- fix(obj) 删除无用的类型转换 (#2343)
- 修复 (lv_obj_scroll.h) 错别字 (#2345)
- 修复（滚动）在两个轴的同一位置触发 LV_EVENT_SCROLL_BEGIN
- 修复 (btnmatrix) 修复焦点变化时按钮失效
- 在单行模式下修复 (textarea) 样式更新 + 改进滚动到光标
- fix(tlsf) 不使用 <assert.h>
- 修复 (imgbtn) 如果只有中间, 请考虑宽度 ==LV_SIZE_CONTENT。img 已设置
- fix(refr) 减少 lv_refr_area 中的嵌套级别
- fix(txt) 增强 break_chars 的功能 (#2327)
- 修复 (pxp): 更新 SDK 2.10 的 RTOS 宏
- 修复 (vglite): v8 更新
- 修复 (pxp): v8 更新
- fix(flex) 修复布局更新和失效问题
- fix(flex) 修复空指针解引用
- fix(obj, switch) 不发送 LV_EVENT_VALUE_CHANGED 两次
- 修复（颜色）溢出 16 位颜色深度
- fix(coords) 使用大坐标修复
- 修复（图表）如果没有添加系列，则修复崩溃
- 使用 LV_CHART_UPDATE_MODE_SHIFT 修复（图表）失效
- 修复（对齐）修复 lv_obj_align_to G
- 修复（表格）使表格单元格值更改无效
- 修复（标签）删除重复的 lv_obj_refresh_self_size
- 修复（绘制）子像素字体绘制中的下溢
- 修复（滚动）不发送不必要的滚动结束事件

11.3 v8.0.1 (14.06.2021)

- docs(filesystem) update to v8 7971ade4
- fix(msgbox) create modals on top layer instead of act screen 5cf6303e
- fix(colowheel) disable LV_OBJ_FLAG_SCROLL_CHAIN by default 48d1c292
- docs(grid) typo fix (#2310) 69d109d2
- fix(arduino) fix the prototype of my_touchpad_read in the LVGL_Arduino.ino 1a62f7a6
- fix(meter) fix needle image invalidation 54d8e817
- fix(mem) add lv_ prefix to tlsf functions and types 0d52b59c
- fix(calendar) fix the position calculation today ad05e196

- fix(typo) rename LV_OBJ_FLAG_SNAPABLE to LV_OBJ_FLAG_SNAPPABLE e697807c
- docs(color) language fixes (#2302) 07ecc9f1
- fix(tick) minor optimization on lv_tick_inc call test b4305df5
- Spelling and other language fixes to documentation (#2293) d0aaacaf
- fix(theme) show disabled state on buttons of btnmatrix, msgbox and keyboard 0be582b3
- fix(scroll) keep the scroll position on object deleted 52edbb46
- fix(msgbox) handle NULL btn map paramter 769c4a30
- fix(group) allow refocusing obejcts 1520208b
- docs(overview) spelling fixes d2efb8c6
- Merge branch 'master' of https://github.com/lvgl/lvgl 45960838
- feat(timer) check if lv_tick_inc is called aa6641a6
- feat(docs) add view on GitHub link a716ac6e
- fix(theme) fix the switch style in the default theme 0c0dc8ea
- docs fix typo 8ab80645
- Merge branch 'master' of https://github.com/lvgl/lvgl e796448f
- feat(event) pass the scroll aniamtion to LV_EVENT_SCROLL_BEGIN ca54ecfe
- fix(tabview) fix with left and right tabs 17c57449
- chore(docs) force docs rebuild 4a0f4139
- chore(docs) always deploy master to docs/master as well 6d05692d
- fix(template) update lv_objx_template to v8 38bb8afc
- docs(extra) add extra/README.md 8cd504d5
- Update CHANGELOG.md 48fd73d2
- Update quick-overview.md (#2295) 5616471c
- fix(pxp) change LV_COLOR_TRANSP to LV_COLOR_CHROMA_KEY to v8 compatibility 81f3068d
- adding micropython examples (#2286) c60ed68e
- docs(color) minor fix ac8f4534
- fix(example) revert test code 77e2c1ff
- fix(draw) with additive blending with 32 bit color depth 786db2af
- docs(color) update colors' docs 9056b5ee
- Merge branch 'master' of https://github.com/lvgl/lvgl a711a1dd
- perf(refresh) optimize where to wait for lv_disp_flush_ready with 2 buffers d0172f14
- docs(lv_obj_style) update add_style and remove_style function headers (#2287) 60f7bcbf
- fix memory leak of spangroup (#2285) 33e0926a
- fix make lv_img_cache.h public becasue cache invalidation is public 38ebcd81
- Merge branch 'master' of https://github.com/lvgl/lvgl 2b292495
- fix(btnmamatrix) fix focus event handling 3b58ef14

- Merge pull request #2280 from lvgl/dependabot/pip/docs/urllib3-1.26.5 a2f45b26
- fix(label) calculating the clip area 57e211cc
- chore(deps): bump urllib3 from 1.26.4 to 1.26.5 in /docs b2f77dfc
- fix(docs) add docs about the default group 29bfe604
- 文档（文件系统）更新到 v8 7971ade4
- 修复（msgbox）在顶层而不是行为屏幕上创建模态 5cf6303e
- 修复（colowheel）默认禁用 LV_OBJ_FLAG_SCROLL_CHAIN 48d1c292
- 文档（网格）错字修复 (#2310) 69d109d2
- fix(arduino) 修复 LVGL_Arduino.ino 中 my_touchpad_read 的原型 1a62f7a6
- fix(meter) 修复针图像失效 54d8e817
- fix(mem) 为 tlf 函数和类型添加 lv_ 前缀 0d52b59c
- fix(calendar) 修正今天的位置计算 ad05e196
- 修复（打字错误）将 LV_OBJ_FLAG_SNAPABLE 重命名为 LV_OBJ_FLAG_SNAPPABLE e697807c
- 文档（颜色）语言修复 (#2302) 07ecc9f1
- 在 lv_tick_inc 调用测试中修复（tick）次要优化 b4305df5
- 文档的拼写和其他语言修复 (#2293) d0aaacaf
- 修复（主题）在 btnmatrix、msgbox 和 keyboard 按钮上显示禁用状态 0be582b3
- 修复（滚动）保持对象上的滚动位置被删除 52edbb46
- 修复（msgbox）处理 NULL btn 映射参数 769c4a30
- fix(group) 允许重新聚焦对象 1520208b
- 文档（概述）拼写修复 d2efb8c6
- 合并 <https://github.com/lvgl/lvgl> 45960838 的分支 “master”
- feat(timer) 检查 lv_tick_inc 是否被调用 aa6641a6
- feat(docs) 在 GitHub 链接上添加视图 a716ac6e
- fix(theme) 修复默认主题中的开关样式 0c0dc8ea
- 文档修复错字 8ab80645
- 合并 <https://github.com/lvgl/lvgl> e796448f 的分支 “master”
- feat(event) 将滚动动画传递给 LV_EVENT_SCROLL_BEGIN ca54ecfe
- fix(tabview) 使用左右标签修复 17c57449
- 杂务 (docs) 强制文档重建 4a0f4139
- 杂务 (docs) 总是将 master 部署到 docs/master 以及 6d05692d
- 修复（模板）更新 lv_objx_template 到 v8 38bb8afc
- docs(extra) 添加 extra/README.md 8cd504d5
- 更新 CHANGELOG.md 48fd73d2
- 更新 quick-overview.md (#2295) 5616471c
- 修复（pxp）将 LV_COLOR_TRANSP 更改为 LV_COLOR_CHROMA_KEY 以兼容 v8 81f3068d

- 添加 micropython 示例 (#2286) c60ed68e
- 文档（颜色）小修正 ac8f4534
- 修复（示例）恢复测试代码 77e2c1ff
- 修复（绘制）与 32 位颜色深度的加法混合 786db2af
- docs(color) 更新颜色的文档 9056b5ee
- 合并 <https://github.com/lvgl/lvgl> a711a1dd 的分支 “master”
- perf(refresh) 使用 2 个缓冲区优化等待 lv_disp_flush_ready 的位置 d0172f14
- docs(lv_obj_style) 更新 add_style 和 remove_style 函数头 (#2287) 60f7bcbf
- 修复 spangroup 的内存泄漏 (#2285) 33e0926a
- 修复使 lv_img_cache.h 公开，因为缓存失效是公开的 38ebcd81
- 合并 <https://github.com/lvgl/lvgl> 的分支 “master” 2b292495
- fix(btnmamatrix) 修复焦点事件处理 3b58ef14
- 从 lvgl/dependabot/pip/docs/urllib3-1.26.5 a2f45b26 合并拉取请求 #2280
- 修复（标签）计算剪辑区域 57e211cc
- 苦差事（deps）：在/docs b2f77dfc 中将 urllib3 从 1.26.4 提升到 1.26.5
- fix(docs) 添加关于默认组的文档 29bfe604

11.4 v8.0.0 (01.06.2021)

v8.0 brings many new features like simplified and more powerful scrolling, new layouts inspired by CSS Flexbox and Grid, simplified and improved widgets, more powerful events, hookable drawing, and more.

v8 is a major change and therefore it's not backward compatible with v7.

v8.0 带来了许多新功能，例如简化且更强大的滚动、受 CSS Flexbox 和 Grid 启发的新布局、简化和改进的小部件、更强大的事件、可挂钩绘图等。

v8 是一个重大变化，因此它不向后兼容 v7。

11.4.1 Directory structure (目录结构)

- The `lv_` prefix is removed from the folder names
- The `docs` is moved to the `lvgl` repository
- The `examples` are moved to the `lvgl` repository
- Create an `src/extrā` folder for complex widgets:
 - It makes the core LVGL leaner
 - In `extrā` we can have a lot and specific widgets
 - Good place for contributions
- 从文件夹名称中删除了 `lv_` 前缀
- `docs` 被移动到 `lvgl` 存储库
- `examples` 被移动到 `lvgl` 存储库

- 为复杂的小部件创建一个 `src/extra` 文件夹:
 - 它使核心 LVGL 更精简
 - 在 `extra` 中，我们可以有很多特定的小部件
 - 贡献的好地方

11.4.2 Widget changes (部件更改)

- `lv_cont` removed, layout features are moved to `lv_obj`
- `lv_page` removed, scroll features are moved to `lv_obj`
- `lv_objmask` the same can be achieved by events
- `lv_meter` added as the union of `lv_linemeter` and `lv_gauge`
- `lv_span` new widget mimicing HTML
- `lv_animating` new widget for simple slideshow animations
- + many minor changes and improvements
- 删除了 `lv_cont`, 布局功能移到了 `lv_obj`
- 删除了 `lv_page`, 滚动功能移到了 `lv_obj`
- `lv_objmask` 同样可以通过事件来实现
- 添加了 `lv_meter` 作为 `lv_linemeter` 和 `lv_gauge` 的联合
- `lv_span` 新小部件模仿 HTML
- `lv_animating` 用于简单幻灯片动画的新小部件
- + 许多小的变化和改进

11.4.3 New scrolling (新的滚动功能)

- Support "elastic" scrolling when scrolled in
- Support scroll chaining among any objects types (not only `lv_pages`)
- Remove `lv_drag`. Similar effect can be achieved by setting the position in `LV_EVENT_PRESSING`
- Add snapping
- Add snap stop to scroll max 1 snap point
- 滚动时支持“弹性”滚动
- 支持任何对象类型之间的滚动链接（不仅是 `lv_pages`）
- 删除了 `lv_drag`。类似的效果可以通过在 `LV_EVENT_PRESSING` 中设置位置来实现
- 添加捕捉
- 添加捕捉停止以滚动最多 1 个捕捉点

11.4.4 New layouts (新的布局)

- CSS Grid-like layout support
- CSS Flexbox-like layout support
- CSS Grid 类似布局支持
- CSS Flexbox 类似布局支持

11.4.5 Styles (样式)

- Optimize and simplify styles
- State is saved in the object instead of the style property
- Object size and position can be set in styles too
- 优化和简化样式
- 状态保存在对象中而不是样式属性中
- 对象大小和位置也可以在样式中设置

11.4.6 Events (事件)

- Allow adding multiple events to an object
- A `user_data` can be attached to the added events
- 允许向一个对象添加多个事件
- 一个 `user_data` 可以附加到添加的事件

11.4.7 Driver changes (驱动程序更改)

- `lv_disp_drv_t`, `lv_indev_drv_t`, `lv_fs_drv_t` needs to be `static`
- ...`disp_buf...` is renamed to `draw_buf`. See an initialization example [here](#).
- No partial update if two screen sized buffers are set
- `disp_drv->full_refresh = 1` makes always the whole display redraw.
- `hor_res` and `ver_res` need to be set in `disp_drv`
- `indev_read_cb` returns `void`. To indicate that there is more that to read set `data->continue_reading = 1` in the `read_cb`
- `lv_disp_drv_t`, `lv_indev_drv_t`, `lv_fs_drv_t` 需要是 `static`
- ...`disp_buf...` 重命名为 `draw_buf`。请参阅 [此处] (https://github.com/lvgl/lv_sim_eclipse_sdl/blob/release/v8.0/main.c#L141) 的初始化示例。
- 如果设置了两个屏幕大小的缓冲区，则不会进行部分更新
- `disp_drv->full_refresh = 1` 总是使整个显示重绘。
- `hor_res` 和 `ver_res` 需要在 `disp_drv` 中设置
- `indev_read_cb` 返回 `void`。表示在 `read_cb` 中还有更多要读取的 set `data->continue_reading = 1`

11.4.8 Other changes (其他变化)

- Remove the copy parameter from create functions
- Simplified File system interface API
- Use a more generic inheritance
- The built-in themes are reworked
- `lv_obj_align` now saves the alignment and realigns the object automatically but can't be used to align to other than the parent
- `lv_obj_align_to` can align to an object but doesn't save the alignment
- `lv_pct(x)` can be used to set the size and position in percentage
- There are many other changes in widgets that are not detailed here. Please refer to the documentation of the widgets.
- 从创建函数中删除复制参数
- 简化的文件系统接口 API
- 使用更通用的继承
- 重新设计了内置主题
- `lv_obj_align` 现在保存对齐并自动重新对齐对象，但不能用于对齐到父对象以外的对象
- `lv_obj_align_to` 可以对齐到一个对象，但不保存对齐
- `lv_pct(x)` 可用于以百分比设置大小和位置
- 小部件中还有许多其他更改，此处未详细说明。请参阅小部件的文档。

11.4.9 New release policy (新的发布政策)

- We will follow Release branches with GitLab flow
- Minor releases are expected in every 3-4 month
- `master` will always contain the latest changes
- 我们将遵循使用 GitLab 流程发布分支
- 预计每 3-4 个月发布一次小版本
- `master` 将始终包含最新的更改

11.4.10 Migrating from v7 to v8 (从 v7 迁移到 v8)

- First and foremost, create a new `lv_conf.h` based on `lv_conf_template.h`.
- To try the new version it's recommended to use a simulator project and see the examples.
- When migrating your project to v8
 - Update the drivers are described above
 - Update the styles
 - Update the events
 - Use the new layouts instead of `lv_cont` features

- Use `lv_obj` instead of `lv_page`
- See the changes in [Colors](#)
- The other parts are mainly minor renames and refactoring. See the functions' documentation for descriptions.
- 首先，基于 `lv_conf_template.h` 创建一个新的 `lv_conf.h`。
- 要尝试新版本，建议使用模拟器项目并查看示例。
- 将项目迁移到 v8 时
 - 更新驱动程序如上所述
 - 更新样式
 - 更新事件
 - 使用新布局而不是 `lv_cont` 功能
 - 使用 `lv_obj` 代替 `lv_page`
 - 查看 [颜色] 中的变化 (<https://docs.lvgl.io/8.0/overview/color.html>)
 - 其他部分主要是小的重命名和重构。有关说明，请参阅函数的文档。

11.5 v7.11.0 (16.03.2021)

11.5.1 New features

- Add better screen orientation management with software rotation support
- Decide text animation's direction based on `base_dir` (when using `LV_USE_BIDI`)

11.5.2 Bugfixes

- fix(gauge) fix needle invalidation
- fix(bar) correct symmetric handling for vertical sliders

11.6 v7.10.1 (16.02.2021)

11.6.1 Bugfixes

- fix(draw) overlap outline with background to prevent aliasing artifacts
- fix(indev) clear the indev's `act_obj` in `lv_indev_reset`
- fix(text) fix out of bounds read in `lv_txt_get_width`
- fix(list) scroll list when button is focused using `LV_KEY_NEXT/PREV`
- fix(text) improve Arabic contextual analysis by adding hyphen processing and proper handling of lam-alef sequence
- fix(delete) delete animation after the children are deleted
- fix(gauge) consider paddings for needle images

11.7 v7.10.0 (02.02.2021)

11.7.1 New features

- feat(indev) allow input events to be passed to disabled objects
- feat(spinbox) add inline get_step function for MicroPython support

11.7.2 Bugfixes

- fix(btnmatrix) fix lv_btnmatrix_get_active_btn_text() when used in a group

11.8 v7.9.1 (19.01.2021)

11.8.1 Bugfixes

- fix(cpicker) fix division by zero
- fix.dropdown) fix selecting options after the last one
- fix(msgbox) use the animation time provided
- fix(gpu_nxp_pxp) fix incorrect define name
- fix(indev) don't leave edit mode if there is only one object in the group
- fix(draw_rect) fix draw pattern stack-use-after-scope error

11.9 v7.9.0 (05.01.2021)

11.9.1 New features

- feat(chart) add lv_chart_remove_series and lv_chart_hide_series
- feat(img_cahce) allow disabling image caching
- calendar: make get_day_of_week() public
- Added support for Zephyr integration

11.9.2 Bugfixes

- fix(draw_rect) free buffer used for arabic processing
- fix(win) arabic process the title of the window
- fix.dropdown) arabic process the option in lv_dropdown_add_option
- fix(textarea) buffer overflow in password mode with UTF-8 characters
- fix(textarea) cursor position after hiding character in password mode
- fix(linemeter) draw critical lines with correct color

- fix(lv_conf_internal) be sure Kconfig defines are always uppercase
- fix(kconfig) handle disable sprintf float correctly.
- fix(layout) stop layout after recursion threshold is reached
- fix(gauge) fix redraw with image needle

11.10 v7.8.1 (15.12.2020)

11.10.1 Bugfixes

- fix(lv_scr_load_anim) fix when multiple screen are loaded at tsame time with delay
- fix(page) fix LV_SCROLLBAR_MODE_DRAG

11.11 v7.8.0 (01.12.2020)

11.11.1 New features

- make DMA2D non blocking
- add unscii-16 built-in font
- add KConfig
- add lv_refr_get_fps_avg()

11.11.2 Bugfixes

- fix(btnmatrix) handle arabic texts in button matrices
- fix(indev) disabled object shouldn't absorb clicks but let the parent to be clicked
- fix(arabic) support processing again already processed texts with _lv_txt_ap_proc
- fix(textarea) support Arabic letter connections
- fix.dropdown) support Arabic letter connections
- fix(value_str) support Arabic letter connections in value string property
- fix(indev) in LV_INDEV_TYPE_BUTTON recognize 1 cycle long presses too
- fix(arc) make arc work with encoder
- fix(slider) adjusting the left knob too with encoder
- fix reference to LV_DRAW_BUF_MAX_NUM in lv_mem.c
- fix(polygon draw) join adjacent points if they are on the same coordinate
- fix(linemeter) fix invalidation when setting new value
- fix(table) add missing invalidation when changing cell type
- refactor(roller) rename LV_ROLLER_MODE_INFINITE -> LV_ROLLER_MODE_INFINITE

11.12 v7.7.2 (17.11.2020)

11.12.1 Bugfixes

- fix(draw_triangle): fix polygon/triangle drawing when the order of points is counter-clockwise
- fix(btnmatrix): fix setting the same map with modified pointers
- fix(arc) fix and improve arc dragging
- label: Repair calculate back `dot` character logical error which cause infinite loop.
- fix(theme_material): remove the bottom border from tabview header
- fix(imgbtn) guess a the closest available state with valid src
- fix(spinbox) update cursor position in lv_spinbox_set_step

11.13 v7.7.1 (03.11.2020)

11.13.1 Bugfixes

- Respect btnmatrix's `one_check` in `lv_btnmatrix_set_btn_ctrl`
- Gauge: make the needle images to use the styles from `LV_GAUGE_PART_PART`
- Group: fix in `lv_group_remove_obj` to handle deleting hidden obejcts correctly

11.14 v7.7.0 (20.10.2020)

11.14.1 New features

- Add PXP GPU support (for NXP MCUs)
- Add VG-Lite GPU support (for NXP MCUs)
- Allow max. 16 cell types for table
- Add `lv_table_set_text_fmt()`
- Use margin on calendar header to set distances and padding to the size of the header
- Add `text_sel_bg` style property

11.14.2 Bugfixes

- Theme update to support text selection background
- Fix imgbtn state change
- Support RTL in table (draw columns right to left)
- Support RTL in pretty layout (draw columns right to left)
- Skip objects in groups if they are in disabled state
- Fix dropdown selection with RTL basedirection

- Fix rectangle border drawing with large width
- Fix `lv_win_clean()`

11.15 v7.6.1 (06.10.2020)

11.15.1 Bugfixes

- Fix BIDI support in dropdown list
- Fix copying base dir in `lv_obj_create`
- Handle sub pixel rendering in font loader
- Fix transitions with style caching
- Fix click focus
- Fix imgbtn image switching with empty style
- Material theme: do not set the text font to allow easy global font change

11.16 v7.6.0 (22.09.2020)

11.16.1 New features

- Check whether any style property has changed on a state change to decide if any redraw is required

11.16.2 Bugfixes

- Fix selection of options with non-ASCII letters in dropdown list
- Fix font loader to support LV_FONT_FMT_TXT_LARGE

11.17 v7.5.0 (15.09.2020)

11.17.1 New features

- Add `clean_dcache_cb` and `lv_disp_clean_dcache` to enable users to use their own cache management function
- Add `gpu_wait_cb` to wait until the GPU is working. It allows to run CPU a wait only when the rendered data is needed.
- Add 10px and 8ox built in fonts

11.17.2 Bugfixes

- Fix unexpected DEFOCUS on lv_page when clicking to bg after the scrollable
- Fix `lv_obj_del` and `lv_obj_clean` if the children list changed during deletion.
- Adjust button matrix button width to include padding when spanning multiple units.
- Add rounding to btnmatrix line height calculation
- Add `decmopr_buf` to GC roots
- Fix division by zero in draw_pattern (lv_draw_rect.c) if the image or letter is not found
- Fix drawing images with 1 px height or width

11.18 v7.4.0 (01.09.2020)

The main new features of v7.4 are run-time font loading, style caching and arc knob with value setting by click.

11.18.1 New features

- Add `lv_font_load()` function - Loads a `lv_font_t` object from a binary font file
- Add `lv_font_free()` function - Frees the memory allocated by the `lv_font_load()` function
- Add style caching to reduce access time of properties with default value
- arc: add set value by click feature
- arc: add `LV_ARC_PART_KNOB` similarly to slider
- send gestures event if the object was dragged. User can check dragging with `lv_indev_is_dragging(lv_indev_act())` in the event function.

11.18.2 Bugfixes

- Fix color bleeding on border drawing
- Fix using 'LV_SCROLLBAR_UNHIDE' after 'LV_SCROLLBAR_ON'
- Fix cropping of last column/row if an image is zoomed
- Fix zooming and rotating mosaic images
- Fix deleting tabview with LEFT/RIGHT tab position
- Fix btnmatrix to not send event when CLICK_TRIG = true and the cursor slid from a pressed button
- Fix roller width if selected text is larger than the normal

11.19 v7.3.1 (18.08.2020)

11.19.1 Bugfixes

- Fix drawing value string twice
- Rename `lv_chart_clear_serie` to `lv_chart_clear_series` and `lv_obj_align_origo` to `lv_obj_align_mid`
- Add linemeter's mirror feature again
- Fix text decor (underline strikethrough) with older versions of font converter
- Fix setting local style property multiple times
- Add missing background drawing and radius handling to image button
- Allow adding extra label to list buttons
- Fix crash if `lv_table_set_col_cnt` is called before `lv_table_set_row_cnt` for the first time
- Fix overflow in large image transformations
- Limit extra button click area of button matrix's buttons. With large paddings it was counter intuitive. (Gaps are mapped to button when clicked).
- Fix `lv_btndmatrix_set_one_check` not forcing exactly one button to be checked
- Fix color picker invalidation in rectangle mode
- Init disabled days to gray color in calendar

11.20 v7.3.0 (04.08.2020)

11.20.1 New features

- Add `lv_task_get_next`
- Add `lv_event_send_refresh`, `lv_event_send_refresh_recursive` to easily send `LV_EVENT_REFRESH` to object
- Add `lv_tabview_set_tab_name()` function - used to change a tab's name
- Add `LV_THEME_MATERIAL_FLAG_NO_TRANSITION` and `LV_THEME_MATERIAL_FLAG_NO_FOCUS` flags
- Reduce code size by adding: `LV_USE_FONT_COMPRESSED` and `LV_FONT_USE_SUBPX` and applying some optimization
- Add `LV_MEMCPY_MEMSET_STD` to use standard `memcpy` and `memset`

11.20.2 Bugfixes

- Do not print warning for missing glyph if its height OR width is zero.
- Prevent duplicated sending of `LV_EVENT_INSERT` from text area
- Tidy outer edges of cpicker widget.
- Remove duplicated lines from `lv_tabview_add_tab`
- btnmatrix: handle combined states of buttons (e.g. checked + disabled)
- textarea: fix typo in `lv_textarea_set_sscrollbar_mode`
- gauge: fix image needle drawing
- fix using freed memory in `_lv_style_list_remove_style`

11.21 v7.2.0 (21.07.2020)

11.21.1 New features

- Add screen transitions with `lv_scr_load_anim()`
- Add display background color, wallpaper and opacity. Shown when the screen is transparent. Can be used with `lv_disp_set_bg_opa/color/image()`.
- Add `LV_CALENDAR_WEEK_STARTS_MONDAY`
- Add `lv_chart_set_x_start_point()` function - Set the index of the x-axis start point in the data array
- Add `lv_chart_set_ext_array()` function - Set an external array of data points to use for the chart
- Add `lv_chart_set_point_id()` function - Set an individual point value in the chart series directly based on index
- Add `lv_chart_get_x_start_point()` function - Get the current index of the x-axis start point in the data array
- Add `lv_chart_get_point_id()` function - Get an individual point value in the chart series directly based on index
- Add `ext_buf_assigned` bit field to `lv_chart_series_t` structure - it's true if external buffer is assigned to series
- Add `lv_chart_set_series_axis()` to assign series to primary or secondary axis
- Add `lv_chart_set_y_range()` to allow setting range of secondary y axis (based on `lv_chart_set_range` but extended with an axis parameter)
- Allow setting different font for the selected text in `lv_roller`
- Add `theme->apply_cb` to replace `theme->apply_xcb` to make it compatible with the MicroPython binding
- Add `lv_theme_set_base()` to allow easy extension of built-in (or any) themes
- Add `lv_obj_align_x()` and `lv_obj_align_y()` functions
- Add `lv_obj_align_origo_x()` and `lv_obj_align_origo_y()` functions

11.21.2 Bugfixes

- `tileview` fix navigation when not screen sized
- Use 14px font by default to for better compatibility with smaller displays
- `linemeter` fix conversation of current value to "level"
- Fix drawing on right border
- Set the cursor image non clickable by default
- Improve mono theme when used with keyboard or encoder

11.22 v7.1.0 (07.07.2020)

11.22.1 New features

- Add `focus_parent` attribute to `lv_obj`
- Allow using buttons in encoder input device
- Add `lv_btnmatrix_set/get_align` capability
- DMA2D: Remove dependency on ST CubeMX HAL
- Added `max_used` property to `lv_mem_monitor_t` struct
- In `lv_init` test if the strings are UTF-8 encoded.
- Add `user_data` to themes
- Add `LV_BIG_ENDIAN_SYSTEM` flag to `lv_conf.h` in order to fix displaying images on big endian systems.
- Add inline function `lv_checkbox_get_state(const lv_obj_t * cb)` to extend the checkbox functionality.
- Add inline function `lv_checkbox_set_state(const lv_obj_t * cb, lv_btn_state_t state)` to extend the checkbox functionality.

11.22.2 Bugfixes

- `lv_img` fix invalidation area when angle or zoom changes
- Update the style handling to support Big endian MCUs
- Change some methods to support big endian hardware.
- remove use of c++ keyword 'new' in parameter of function `lv_theme_set_base()`.
- Add `LV_BIG_ENDIAN_SYSTEM` flag to `lv_conf.h` in order to fix displaying images on big endian systems.
- Fix inserting chars in text area in big endian hardware.

11.23 v7.0.2 (16.06.2020)

11.23.1 Bugfixes

- `lv_textarea` fix wrong cursor position when clicked after the last character
- Change all text related indices from 16-bit to 32-bit integers throughout whole library. #1545
- Fix gestures
- Do not call `set_px_cb` for transparent pixel
- Fix list button focus in material theme
- Fix crash when the a text area is cleared with the backspace of a keyboard
- Add version number to `lv_conf_template.h`
- Add log in true double buffering mode with `set_px_cb`
- `lv_dropdown`: fix missing `LV_EVENT_VALUE_CHANGED` event when used with encoder
- `lv_tileview`: fix if not the {0;0} tile is created first
- `lv_debug`: restructure to allow asserting in from `lv_misc` too
- add assert if `_lv_mem_buf_get()` fails
- `lv_textarea`: fix character delete in password mode
- Update `LV_OPA_MIN` and `LV_OPA_MAX` to widen the opacity processed range
- `lv_btm` fix sending events for hidden buttons
- `lv_gauge` make `lv_gauge_set_angle_offset` offset the labels and needles too
- Fix typo in the API `scrlable -> scrollable`
- `tabview` by default allow auto expanding the page only to right and bottom (#1573)
- fix crash when drawing gradient to the same color
- chart: fix memory leak
- `img`: improve hit test for transformed images

11.24 v7.0.1 (01.06.2020)

11.24.1 Bugfixes

- Make the MicroPython working by adding the required variables as `GC_ROOT`
- Prefix some internal API functions with `_` to reduce the API of LVGL
- Fix built-in SimSun CJK font
- Fix UTF-8 encoding when `LV_USE_ARABIC_PERSIAN_CHARS` is enabled
- Fix DMA2D usage when 32 bit images directly blended
- Fix `lv_roller` in infinite mode when used with encoder
- Add `lv_theme_get_color_secondary()`

- Add `LV_COLOR_MIX_ROUND_OFS` to adjust color mixing to make it compatible with the GPU
- Improve DMA2D blending
- Remove `memcpy` from `lv_ll` (caused issues with some optimization settings)
- `lv_chart` fix X tick drawing
- Fix vertical dashed line drawing
- Some additional minor fixes and formatings

11.25 v7.0.0 (18.05.2020)

11.25.1 Documentation

The docs for v7 is available at <https://docs.littlevgl.com/v7/en/html/index.html>

11.25.2 Legal changes

The name of the project is changed to LVGL and the new website is on <https://lvgl.io>

LVGL remains free under the same conditions (MIT license) and a company is created to manage LVGL and offer services.

11.25.3 New drawing system

Complete rework of LVGL's draw engine to use "masks" for more advanced and higher quality graphical effects. A possible use-case of this system is to remove the overflowing content from the rounded edges. It also allows drawing perfectly anti-aliased circles, lines, and arcs. Internally, the drawings happen by defining masks (such as rounded rectangle, line, angle). When something is drawn the currently active masks can make some pixels transparent. For example, rectangle borders are drawn by using 2 rectangle masks: one mask removes the inner part and another the outer part.

The API in this regard remained the same but some new functions were added:

- `lv_img_set_zoom`: set image object's zoom factor
- `lv_img_set_angle`: set image object's angle without using canvas
- `lv_img_set_pivot`: set the pivot point of rotation

The new drawing engine brought new drawing features too. They are highlighted in the "style" section.

11.25.4 New style system

The old style system is replaced with a new more flexible and lightweighted one. It uses an approach similar to CSS: support cascading styles, inheriting properties and local style properties per object. As part of these updates, a lot of objects were reworked and the APIs have been changed.

- more shadows options: *offset* and *spread*
- gradient stop position to shift the gradient area and horizontal gradient
- `LV_BLEND_MODE_NORMAL/ADDITIVE/SUBTRACTIVE` blending modes
- `clip corner`: crop the content on the rounded corners
- `text underline` and `strikethrough`

- dashed vertical and horizontal lines (*dash gap*, *dash width*)
- *outline*: a border-like part drawn out of the background. Can have spacing to the background.
- *pattern*: display and image in the middle of the background or repeat it
- *value* display a text which is stored in the style. It can be used e.g. as a lighweighted text on buttons too.
- *margin*: similar to *padding* but used to keep space outside of the object

Read the [Style](#) section of the documentation to learn how the new styles system works.

11.25.5 GPU integration

To better utilize GPUs, from this version GPU usage can be integrated into LVGL. In `lv_conf.h` any supported GPUs can be enabled with a single configuration option.

Right now, only ST's DMA2D (Chrom-ART) is integrated. More will in the upcoming releases.

11.25.6 Renames

The following object types are renamed:

- sw -> switch
- ta -> textarea
- cb -> checkbox
- lmeter -> linemeter
- mbox -> msgbox
- dlist -> dropdown
- btnm -> btnmatrix
- kb -> keyboard
- preload -> spinner
- lv_objx folder -> lv_widgets
- LV_FIT_FILL -> LV_FIT_PARENT
- LV_FIT_FLOOD -> LV_FLOOD_MAX
- LV_LAYOUT_COL_L/M/R -> LV_LAYOUT_COLUMN_LEFT/MID/RIGHT
- LV_LAYOUT_ROW_T/M/B -> LV_LAYOUT_ROW_TOP/MID/BOTTOM

11.25.7 Reworked and improved object

- **dropdown**: Completely reworked. Now creates a separate list when opened and can be dropped to down/up/left/right.
- **label**: `body_draw` is removed, instead, if its style has a visible background/border/shadow etc it will be drawn. Padding really makes the object larger (not just virtually as before)
- **arc**: can draw bacground too.
- **btn**: doesn't store styles for each state because it's done naturally in the new style system.

- **calendar**: highlight the pressed datum. The used styles are changed: use `LV_CALENDAR_PART_DATE` normal for normal dates, checked for highlighted, focused for today, pressed for the being pressed. (checked+pressed, focused+pressed also work)
- **chart**: only has `LINE` and `COLUMN` types because with new styles all the others can be described. `LV_CHART_PART_SERIES` sets the style of the series. `bg_opa > 0` draws an area in `LINE` mode. `LV_CHART_PART_SERIES_BG` also added to set a different style for the series area. Padding in `LV_CHART_PART_BG` makes the series area smaller, and it ensures space for axis labels/numbers.
- **linemeter, gauge**: can have background if the related style properties are set. Padding makes the scale/lines smaller. `scale_border_width` and `scale_end_border_width` allow to draw an arc on the outer part of the scale lines.
- **gauge**: `lv_gauge_set_needle_img` allows use image as needle
- **canvas**: allow drawing to true color alpha and alpha only canvas, add `lv_canvas_blur_hor/ver` and rename `lv_canvas_rotate` to `lv_canvas_transform`
- **textarea**: If available in the font use bullet (U+2022) character in text area password

11.25.8 New object types

- `lv_objmask`: masks can be added to it. The children will be masked accordingly.

11.25.9 Others

- Change the built-in fonts to `Montserrat` and add built-in fonts from 12 px to 48 px for every 2nd size.
- Add example CJK and Arabic/Persian/Hebrew built-in font
- Add ° and "bullet" to the built-in fonts
- Add Arabic/Persian script support: change the character according to its position in the text.
- Add `playback_time` to animations.
- Add `repeat_count` to animations instead of the current "repeat forever".
- Replace `LV_LAYOUT_PRETTY` with `LV_LAYOUT_PRETTY_TOP/MID/BOTTOM`

11.25.10 Demos

- `lv_examples` was reworked and new examples and demos were added

11.25.11 New release policy

- Maintain this Changelog for every release
- Save old major version in new branches. E.g. `release/v6`
- Merge new features and fixes directly into `master` and release a patch or minor releases every 2 weeks.

11.25.12 Migrating from v6 to v7

- First and foremost, create a new `lv_conf.h` based on `lv_conf_template.h`.
- To try the new version it suggested using a simulator project and see the examples.
- If you have a running project, the most difficult part of the migration is updating to the new style system. Unfortunately, there is no better way than manually updating to the new format.
- The other parts are mainly minor renames and refactoring as described above.

CHAPTER 12

Roadmap (产品线路)

This is a summary for planned new features and a collection of ideas. This list indicates only the current intention and it can be changed.

这是计划中的新功能的摘要和想法的集合。此列表仅表示当前意图，并且可以更改。

12.1 v8.1

12.1.1 Features (功能)

- [x] Unit testing (gtest?). See #1658
- [] Benchmarking (gem5 or qemu?). See #1660
- [] lv_snapshot: buffer a widget and all of its children into an image. The source widget can be on a different screen too. The resulting image can be transformed.
- [] High level GPU support. See #2058
- [x] 单元测试 (gtest?)。见 #1658
- [] 基准测试 (gem5 或 qemu?)。见 #1660
- [] lv_snapshot: 将小部件及其所有子部件缓冲到图像中。源小部件也可以在不同的屏幕上。可以转换生成的图像。
- [] 高级 GPU 支持。见 #2058

New features (新功能)

- [x] merge MicroPython examples
- [x] add a "Try out yourself" button to the Micropython examples
- [x] 合并 MicroPython 示例
- [x] 在 Micropython 示例中添加“试用自己”按钮

12.1.2 Discuss (讨论)

- [] CPP binding
- [] Plugins. In v8 core and extra widgets are separated. With the new flexible events, the behavior of the widgets can be modified in a modular way. E.g. a plugin to add faded area to a line chart (as in the widgets demo)
- [] CPP 绑定
- [] 插件。在 v8 核心和额外的小部件是分开的。使用新的灵活事件，可以以模块化方式修改小部件的行为。例如。将褪色区域添加到折线图的插件（如小部件演示中所示）

12.1.3 Docs (文档)

- [x] Display the Micropython examples too.
- [x] Add a link to the example C and py files
- [x] List of all examples on a page. All in iframes grouped by category (e.g. flex, style, button)
- [x] 也显示 Micropython 示例。
- [x] 添加指向示例 C 和 py 文件的链接
- [x] 页面上所有示例的列表。按类别分组的所有 iframe（例如 flex、样式、按钮）

12.1.4 Others (其他)

- [] Add automatic rebuild to get binary directly. Similarly to STM32F746 project.
- [] Implement release scripts. I've added a basic specification [here](#), but we should discuss it.
- [] Unit test for the core widgets
- [] 添加自动重建直接获取二进制文件。类似于 STM32F746 项目。
- [] 实现发布脚本。我已经添加了一个基本规范 [here](#)，但我们应该讨论它。
- [] 核心小部件的单元测试

12.2 v8.2

- [] Optimize line and circle drawing and masking
- [] Handle stride. See #1858
- [] Support LV_STATE_HOVERED
- [] 优化线和圆的绘制和遮罩
- [] 处理步幅。见 #1858
- [] 支持 LV_STATE_HOVERED

12.3 Ideas (想法)

- Reconsider color format management for run time color format setting, and custom color format usage. (Also [RGB888](#))
- Make gradients more versatile
- Make image transformations more versatile
- Switch to RGBA colors in styles
- Consider direct binary font format support
- Simplify [groups](#). Discussion is [here](#).
- Use [generate-changelog](#) to automatically generate changelog
- `lv_mem_alloc_aligned(size, align)`
- Text node. See #1701
- CPP binding. See [Forum](#)
- Optimize font decompression
- Need coverage report for tests
- Need static analyze (via coverity.io or something else)
- Support dot_begin and dot_middle long modes for labels
- Add new label alignment modes. #1656
- Support larger images: #1892

-
- 重新考虑运行时颜色格式设置和自定义颜色格式使用的颜色格式管理。(还有 [RGB888](#))
 - 使渐变更加通用
 - 使图像转换更加通用
 - 在样式中切换到 RGBA 颜色
 - 考虑直接二进制字体格式支持
 - 简化 [groups](#)。讨论在 [这里] (<https://forum.lvgl.io/t/lv-group-tabindex/2927/3>)。
 - 使用[generate-changelog](#)自动生成 changelog
 - `lv_mem_alloc_aligned` (大小, 对齐)

- 文本节点。见#1701
 - CPP 绑定。见论坛
 - 优化字体解压
 - 需要测试覆盖率报告
 - 需要静态分析（通过 coverity.io 或其他）
 - 支持 dot_begin 和 dot_middle 长标签模式
 - 添加新的标签对齐模式。#1656
 - 支持更大的图像：#1892
-

12.4 v8

- Create an `extra` folder for complex widgets
 - It makes the core LVGL leaner
 - In `extra` we can have a lot and specific widgets
 - Good place for contributions
- New scrolling:
 - See `feat/new-scroll` branch and #1614) issue.
 - Remove `lv_page` and support scrolling on `lv_obj`
 - Support "elastic" scrolling when scrolled in
 - Support scroll chaining among any objects types (not only `lv_pages`)
 - Remove `lv_drag`. Similar effect can be achieved by setting the position in `LV_EVENT_Pressing`
 - Add snapping
 - Add snap stop to scroll max 1 snap point
 - Already working
- New layouts:
 - See #1615 issue
 - CSS Grid-like layout support
 - CSS Flexbox-like layout support
 - Remove `lv_cont` and support layouts on `lv_obj`
- Simplified File system interface (`feat/new_fs_api` branch) to make porting easier
 - Work in progress
- Remove the align parameter from `lv_canvas_draw_text`
- Remove the copy parameter from create functions
- Optimize and simplifie styles #1832
- Use a more generic inheritenace #1919

- Allow adding multiple events to an obejct
- 为复杂的小部件创建一个 `extra` 文件夹
 - 它使核心 LVGL 更精简
 - 在 `extra` 中，我们可以有很多特定的小部件
 - 贡献的好地方
- 新滚动：
 - 参见 `feat/new-scroll` 分支和 [#1614](<https://github.com/lvgl/lvgl/issues/1614>) 问题。
 - 移除 `lv_page` 并支持在 `lv_obj` 上滚动
 - 滚动时支持“弹性”滚动
 - 支持任何对象类型之间的滚动链接（不仅是 `lv_pagess`）
 - 删除 `lv_drag`。类似的效果可以通过在 `LV_EVENT_Pressing` 中设置位置来实现
 - 添加捕捉
 - 添加捕捉停止以滚动最多 1 个捕捉点
 - 已经工作
- 新布局：
 - 参见 #1615 问题
 - `CSS Grid` 类似布局支持
 - `CSS Flexbox` 类似布局支持
 - 删除 `lv_cont` 并支持 `lv_obj` 上的布局
- 简化的文件系统接口 (`[feat/new_fs_api]` (<https://github.com/lvgl/lvgl/tree/feat/new-fs-api>) 分支) 使移植更容易
 - 工作正在进行中
- 从 `lv_canvas_draw_text` 中删除 `align` 参数
- 从创建函数中删除复制参数
- 优化和简化样式 #1832
- 使用更通用的继承 #1919
- 允许向一个对象添加多个事件

④ 项目实战

13.1 在 windows 模拟器运行 lvgl(v8.0)

13.1.1 Code::Blocks 上运行

Code::Blocks 是一个免费开放源码的全功能的跨平台 C/C++ 集成开发环境。使用 Code::Blocks 模拟器体验或开发 lvgl，开箱即用很方便，相比 VS 更加轻量级。

获取资料

- 百度云下载：链接：https://pan.baidu.com/s/14renDMjP6xBVh0bGy_yAWA 提取码：root

获取 Code::Blocks 并安装



使用了当前的最新版本 20.03。

文件 > ... > 01_windows平台 > 01_codeblocks > 02_软件

○ 共 1 项



软件安装包在资料中的这个位置：

下载之后直接打开即可安装，安装过程按照软件提示进行安装即可，最后启动并打开 Code::Blocks，下一步准备通过 Code::Blocks 打开一个 lvgl 示例工程。

<https://blog.csdn.net/qd>

获取示例源码并运行

下载资料，从这里获取 lvgl 示例工程源码：

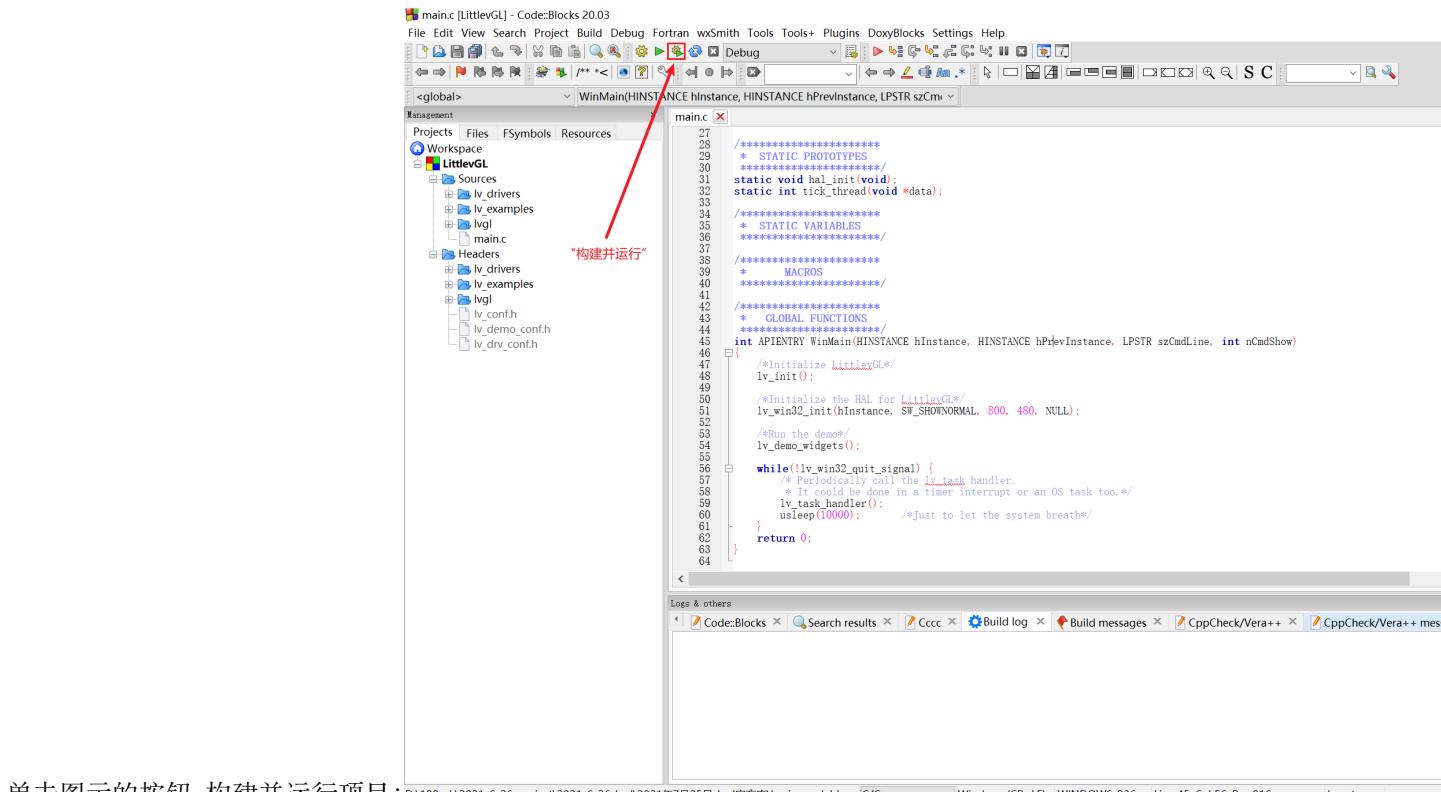


https://blog.csdn.net/qq_35181236

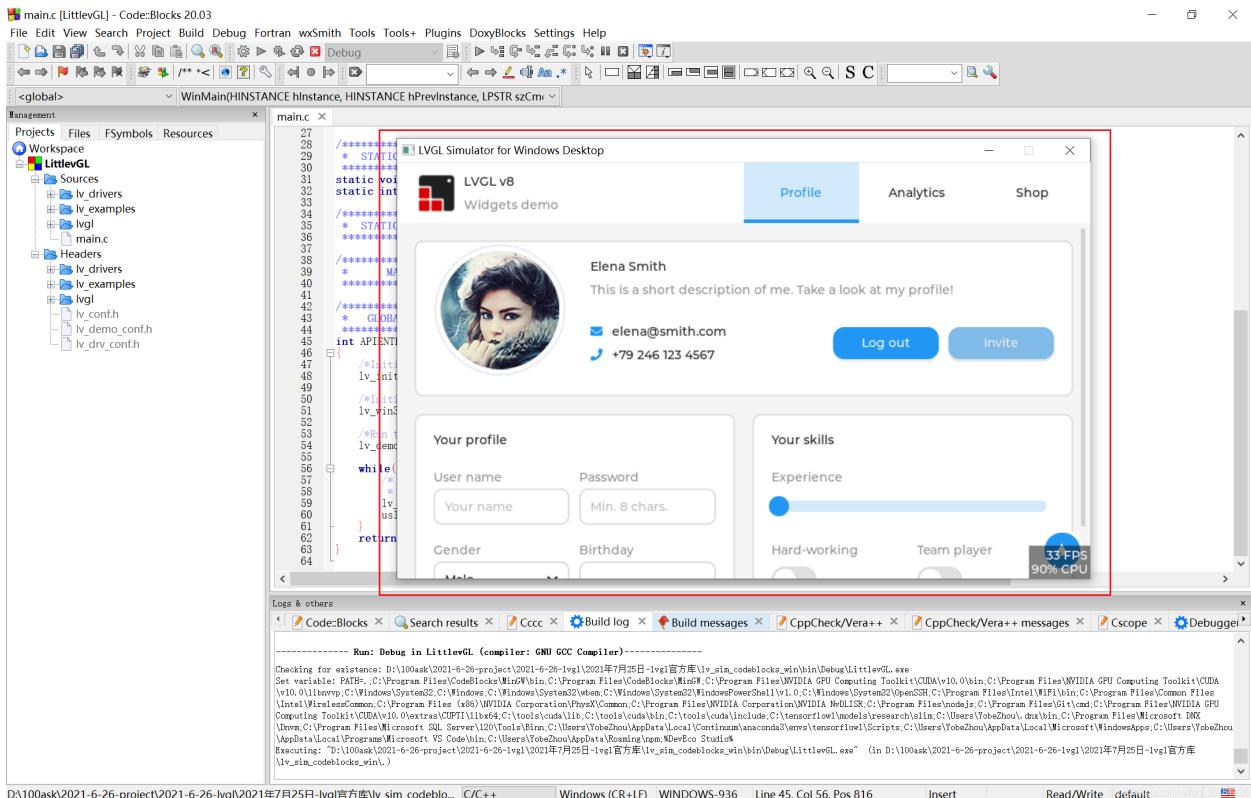
解压后，进入目录双击 LittlevGL.cbp 可直接打开项目工程：

名称	修改日期	类型
.git	2021/7/25 22:23	文件夹
.github	2021/7/25 22:22	文件夹
lv_drivers	2021/7/25 22:37	文件夹
lv_examples	2021/7/25 22:37	文件夹
lvgl	2021/7/25 22:37	文件夹
.gitignore	2021/7/25 22:22	文本文档
.gitmodules	2021/7/25 22:22	文本文档
licence.txt	2021/7/25 22:22	TXT 文件
LICENSE	2021/7/25 22:22	文件
LittlevGL.cbp	2021/7/25 22:22	CBP 文件
LittlevGL.layout	2021/7/25 22:22	LAYOUT 文件
lv_conf.h	2021/7/25 22:22	H 文件
lv_demo_conf.h	2021/7/25 22:22	H 文件
lv_drv_conf.h	2021/7/25 22:22	H 文件
lvgl_icon.bmp	2021/7/25 22:22	BMP 文件
main.c	2021/7/25 22:22	C 文件
README.md	2021/7/25 22:22	Markdown 源文

< https://blog.csdn.net/qq_35181239 >



单击图示的按钮，构建并运行项目：D:\100ask\2021-6-26-project\2021-6-26-lvgl\2021年7月25日-lvgl\官方库\lv_sim_codeblo... C/C++ Windows (CR+LF) WINDOWS-936 Line 45, Col 56, Pos 816 Insert



尽情享受 LVGL 带来的惊喜吧！

13.1. 在 windwos 模拟器运行 lvgl(v8.0)

13.1.2 在 Eclipse 上运行

TODO

13.1.3 在 vscode 上运行

TODO

13.2 STM32F103 LVGL GUI DEMO 效果



已实现了以下功能：

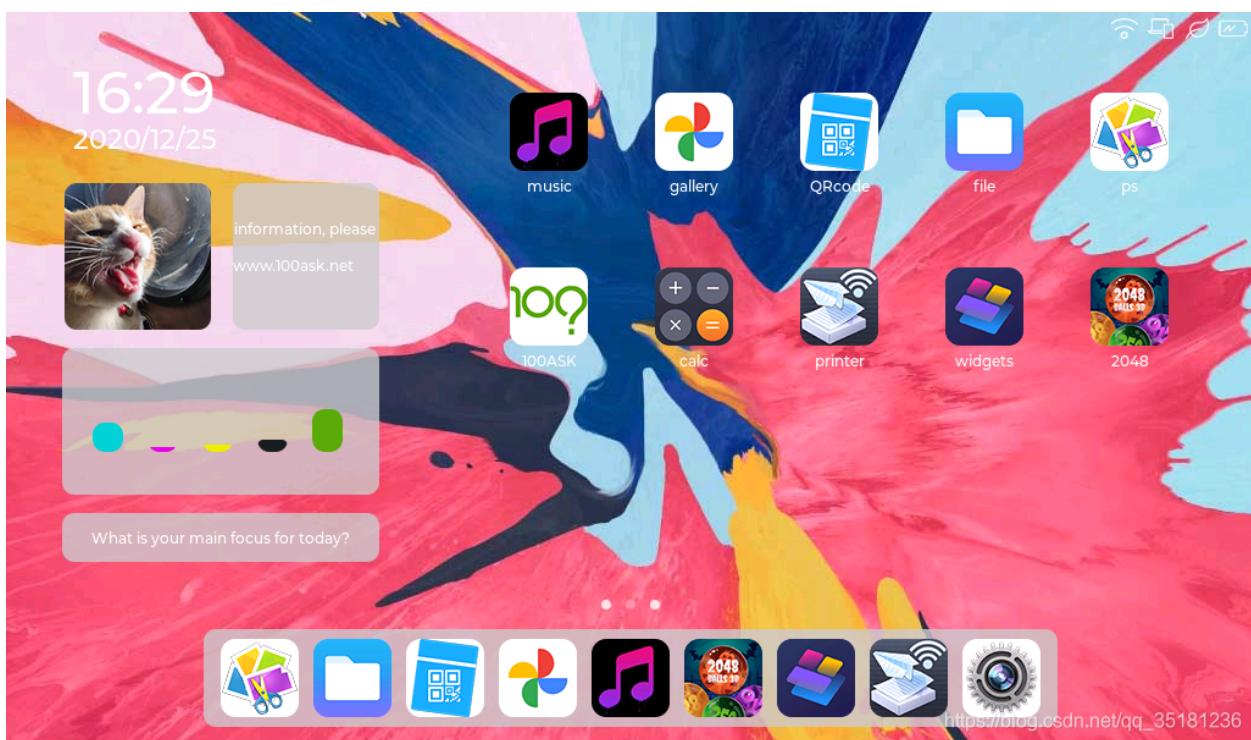
- 模仿 windows10 风格的文件浏览器
- 时钟
- 二维码生成器
- 系统说明
- 温湿度采集
- 2048 小游戏
- 音乐播放器
- 贪吃蛇小游戏
- 计算器
- 寄存器位查看工具

- 系统主题切换
- 板载硬件测试

13.2.1 STM32F103 LVGL GUI DEMO 源码

stm32f103 源码 git 仓库 ([点击跳转](#))

13.3 STM32MP157 LVGL GUI DEMO 效果



- 图库
- 二维码生成器
- 文件浏览器
- 集成 lvgl 官方 demo: 图片编辑器
- 集成 lvgl 官方 demo: 音乐播放器
- 集成 lvgl 官方 demo: 打印机
- 集成 lvgl 官方 demo: 压力测试
- 2048 小游戏
- TODO...

13.3.1 STM32MP157 LVGL GUI DEMO 源码

stm32mp157-lvgl 源码 git 仓库 ([点击跳转](#))

stm32mp157-buildroot 源码 git 仓库 ([点击跳转](#))

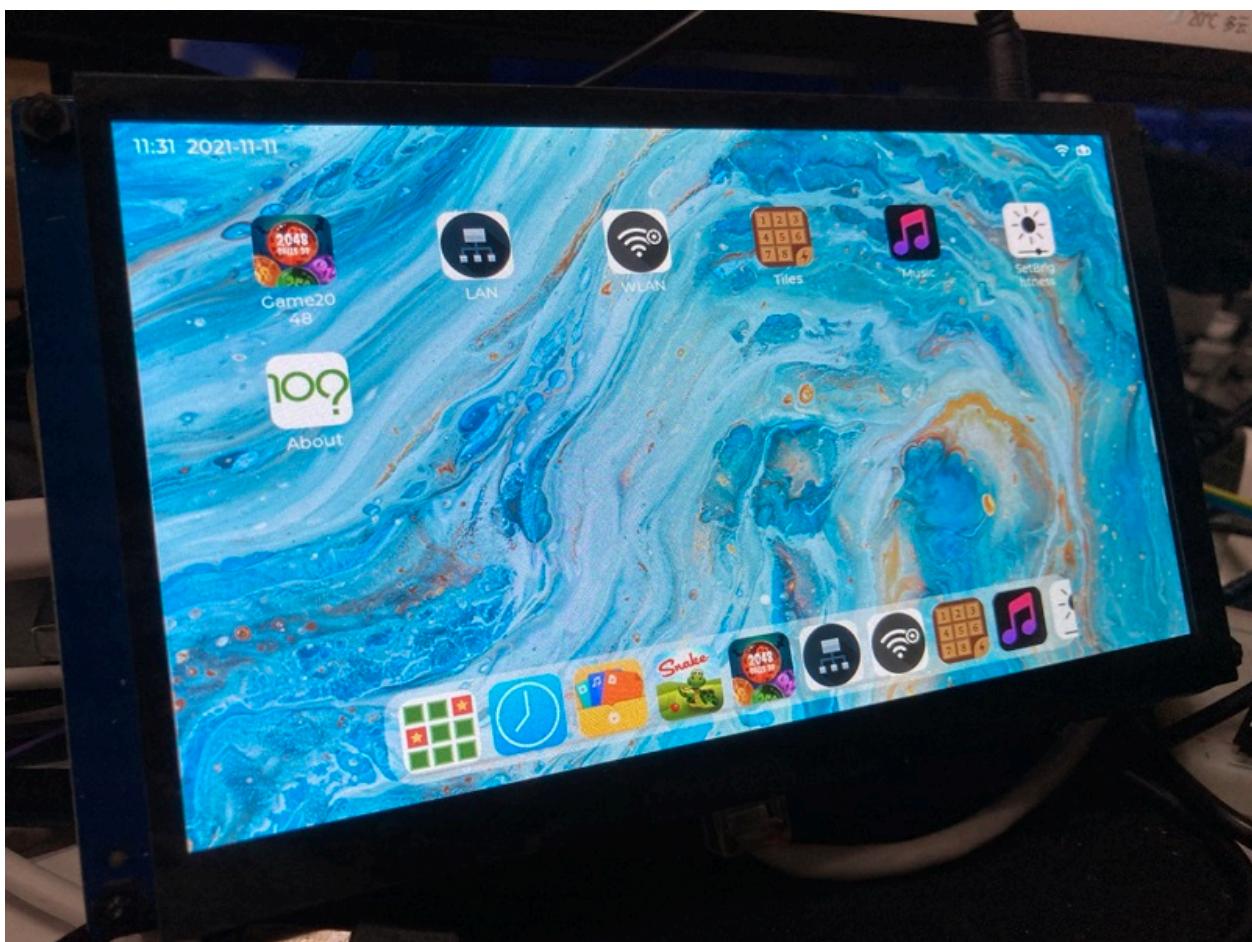
13.4 IMX6ULL LVGL GUI DEMO 效果

13.4.1 IMX6ULL Linux LVGL GUI V2.0

Linux lvgl gui 2.0 和大家见面啦！

- 全新的架构，功能更强大
- 二次开发非常方便
- 独立的应用之间使用 dbus 通信
- GUI 基于 lvgl 8.1 开发，长期更新支持 lvgl 8.x
- 还有更多细节等你来探索！
- ...

演示视频：<https://www.bilibili.com/video/BV1nT4y1R7rz>



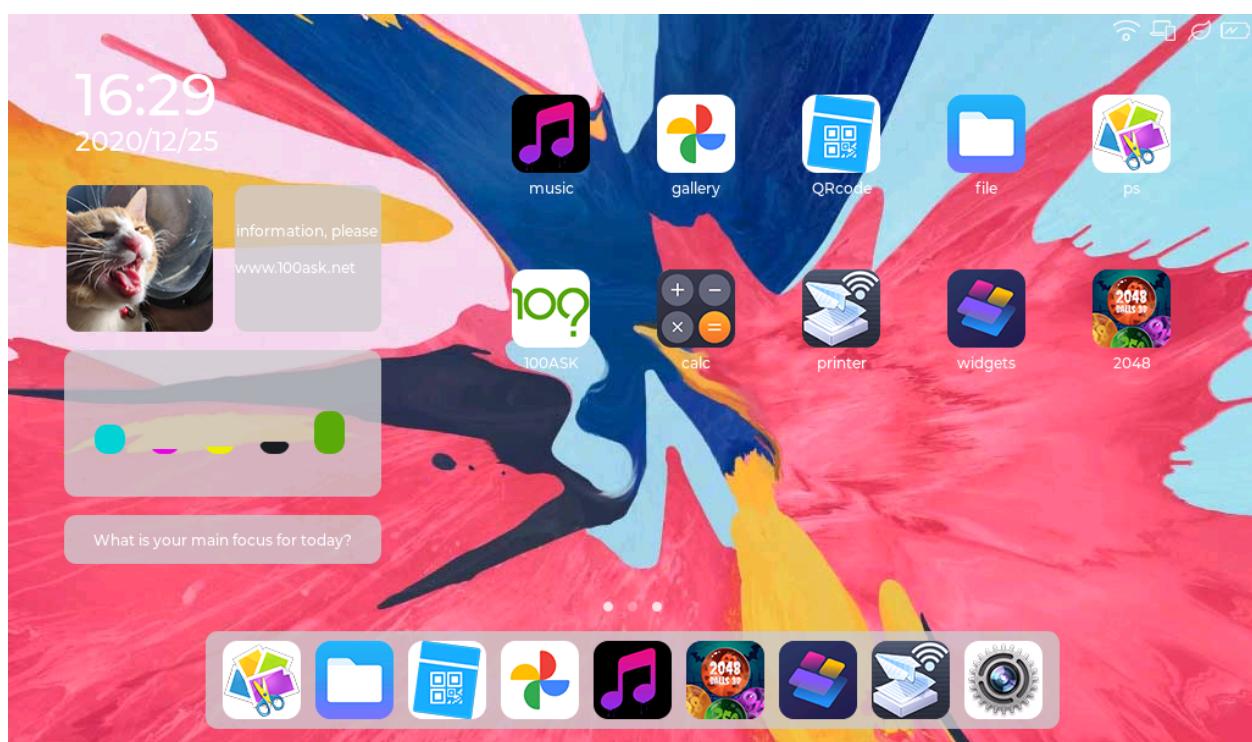
源码

IMX6ULL Linux LVGL GUI V2.0 源码 git 仓库 ([点击跳转](#))

13.4.2 IMX6ULL Linux LVGL GUI 1.0

- 图库
- 二维码生成器
- 文件浏览器
- 集成 lvgl 官方 demo: 图片编辑器
- 集成 lvgl 官方 demo: 音乐播放器
- 集成 lvgl 官方 demo: 打印机
- 集成 lvgl 官方 demo: 压力测试
- 2048 小游戏
- TODO...

演示视频: <https://www.bilibili.com/video/BV1cU4y1b7dY>



源码

IMX6ULL Linux LVGL GUI 1.0 源码 git 仓库 ([点击跳转](#))

13.5 ESP32 LVGL GUI DEMO 效果

TODO

CHAPTER 14

联系我们

- 【1】韦东山老师官方在线学习平台
- 【2】单片机工程师如何提升自己进阶嵌入式 Linux?
- 【3】访问 B 站主页查看更多精彩内容
- 【4】韦东山老师 CSDN 博客
- 【5】打开手机微信扫一扫，关注公众号 【百问科技】，获取更多嵌入式系统干货文章



- 【6】打开手机微信扫一扫，微信学习小程序学习更多课程！



CHAPTER 15

加入技术交流群聊一起学习！

符号

`_lv_anim_core_init(C++ function)`, 257
`_lv_anim_t(C++ struct)`, 261
`_lv_anim_t::act_time(C++ member)`, 262
`_lv_anim_t::current_value (C++ member)`,
 262
`_lv_anim_t::early_apply(C++ member)`, 262
`_lv_anim_t::end_value(C++ member)`, 262
`_lv_anim_t::exec_cb(C++ member)`, 261
`_lv_anim_t::get_value_cb (C++ member)`, 261
`_lv_anim_t::path_cb(C++ member)`, 261
`_lv_anim_t::playback_delay (C++ member)`,
 262
`_lv_anim_t::playback_now(C++ member)`, 262
`_lv_anim_t::playback_time (C++ member)`,
 262
`_lv_anim_t::ready_cb(C++ member)`, 261
`_lv_anim_t::repeat_cnt(C++ member)`, 262
`_lv_anim_t::repeat_delay(C++ member)`, 262
`_lv_anim_t::run_round(C++ member)`, 262
`_lv_anim_t::start_cb(C++ member)`, 261
`_lv_anim_t::start_cb_called(C++ member)`,
 262
`_lv_anim_t::start_value(C++ member)`, 261
`_lv_anim_t::time(C++ member)`, 262
`_lv_anim_t::time_orig(C++ member)`, 262
`_lv_anim_t::user_data(C++ member)`, 261
`_lv_anim_t::var(C++ member)`, 261
`_lv_color_filter_dsc_t(C++ struct)`, 216
`_lv_color_filter_dsc_t::filter_cb (C++
 member)`, 217
`_lv_color_filter_dsc_t::user_data (C++
 member)`, 217
`_lv_disp_drv_t(C++ struct)`, 95
`_lv_disp_drv_t::antialiasing (C++ mem-
 ber)`, 96
`_lv_disp_drv_t::clean_dcache_cb (C++
 member)`, 96
`_lv_disp_drv_t::color_chroma_key (C++
 member)`, 97

`_lv_disp_drv_t::dpi (C++ member)`, 96
`_lv_disp_drv_t::draw_buf (C++ member)`, 95
`_lv_disp_drv_t::drv_update_cb (C++ mem-
 ber)`, 96
`_lv_disp_drv_t::flush_cb (C++ member)`, 96
`_lv_disp_drv_t::full_refresh (C++ mem-
 ber)`, 95
`_lv_disp_drv_t::gpu_fill_cb (C++ member)`,
 97
`_lv_disp_drv_t::gpu_wait_cb (C++ member)`,
 96
`_lv_disp_drv_t::hor_res (C++ member)`, 95
`_lv_disp_drv_t::monitor_cb (C++ member)`,
 96
`_lv_disp_drv_t::rotated (C++ member)`, 96
`_lv_disp_drv_t::rounder_cb (C++ member)`,
 96
`_lv_disp_drv_t::screen_transp (C++ mem-
 ber)`, 96
`_lv_disp_drv_t::set_px_cb (C++ member)`, 96
`_lv_disp_drv_t::sw_rotate (C++ member)`, 96
`_lv_disp_drv_t::user_data (C++ member)`, 97
`_lv_disp_drv_t::ver_res (C++ member)`, 95
`_lv_disp_drv_t::wait_cb (C++ member)`, 96
`_lv_disp_get_refr_timer(C++ function)`, 207
`_lv_disp_t(C++ struct)`, 97
`_lv_disp_t::act_scr (C++ member)`, 97
`_lv_disp_t::bg_color (C++ member)`, 98
`_lv_disp_t::bg_img (C++ member)`, 98
`_lv_disp_t::bg_opa (C++ member)`, 98
`_lv_disp_t::del_prev (C++ member)`, 97
`_lv_disp_t::driver (C++ member)`, 97
`_lv_disp_t::inv_area_joined (C++ member)`,
 98
`_lv_disp_t::inv_areas (C++ member)`, 98
`_lv_disp_t::inv_p (C++ member)`, 98
`_lv_disp_t::last_activity_time (C++ mem-
 ber)`, 98
`_lv_disp_t::prev_scr (C++ member)`, 97

`_lv_disp_t::refr_timer(C++ member)`, 97
`_lv_disp_t::scr_to_load(C++ member)`, 97
`_lv_disp_t::screen_cnt(C++ member)`, 97
`_lv_disp_t::screens(C++ member)`, 97
`_lv_disp_t::sys_layer(C++ member)`, 97
`_lv_disp_t::theme(C++ member)`, 97
`_lv_disp_t::top_layer(C++ member)`, 97
`_lv_fs_drv_t(C++ struct)`, 251
`_lv_fs_drv_t::close_cb(C++ member)`, 251
`_lv_fs_drv_t::dir_close_cb(C++ member)`,
 251
`_lv_fs_drv_t::dir_open_cb(C++ member)`,
 251
`_lv_fs_drv_t::dir_read_cb(C++ member)`,
 251
`_lv_fs_drv_t::letter(C++ member)`, 251
`_lv_fs_drv_t::open_cb(C++ member)`, 251
`_lv_fs_drv_t::read_cb(C++ member)`, 251
`_lv_fs_drv_t::ready_cb(C++ member)`, 251
`_lv_fs_drv_t::seek_cb(C++ member)`, 251
`_lv_fs_drv_t::tell_cb(C++ member)`, 251
`_lv_fs_drv_t::user_data(C++ member)`, 251
`_lv_fs_drv_t::write_cb(C++ member)`, 251
`_lv_fs_init(C++ function)`, 249
`_lv_group_init(C++ function)`, 199
`_lv_group_t(C++ struct)`, 201
`_lv_group_t::editing(C++ member)`, 201
`_lv_group_t::focus_cb(C++ member)`, 201
`_lv_group_t::frozen(C++ member)`, 201
`_lv_group_t::obj_focus(C++ member)`, 201
`_lv_group_t::obj_ll(C++ member)`, 201
`_lv_group_t::refocus_policy(C++ member)`,
 201
`_lv_group_t::user_data(C++ member)`, 201
`_lv_group_t::wrap(C++ member)`, 201
`_lv_img_buf_get_transformed_area(C++
 function)`, 242
`_lv_img_buf_transform(C++ function)`, 241
`_lv_img_buf_transform_anti_alias(C++
 function)`, 241
`_lv_img_buf_transform_init(C++ function)`,
 241
`_lv_indev_drv_t(C++ struct)`, 106
`_lv_indev_drv_t::disp(C++ member)`, 106
`_lv_indev_drv_t::feedback_cb(C++ mem-
 ber)`, 106
`_lv_indev_drv_t::gesture_limit(C++ mem-
 ber)`, 106
`_lv_indev_drv_t::gesture_min_velocity
 (C++ member)`, 106
`_lv_indev_drv_t::long_press_repeat_time
 (C++ member)`, 106
`_lv_indev_drv_t::long_press_time(C++
 member)`, 106
`_lv_indev_drv_t::read_cb(C++ member)`, 106
`_lv_indev_drv_t::read_timer(C++ member)`,
 106
`_lv_indev_drv_t::scroll_limit(C++ mem-
 ber)`, 106
`_lv_indev_drv_t::scroll_throw(C++ mem-
 ber)`, 106
`_lv_indev_drv_t::type(C++ member)`, 106
`_lv_indev_drv_t::user_data(C++ member)`,
 106
`_lv_indev_proc_t(C++ struct)`, 106
`_lv_indev_proc_t::act_obj(C++ member)`,
 107
`_lv_indev_proc_t::act_point(C++ member)`,
 107
`_lv_indev_proc_t::disabled(C++ member)`,
 107
`_lv_indev_proc_t::gesture_dir(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::gesture_sent(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::gesture_sum(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::keypad(C++ member)`, 107
`_lv_indev_proc_t::last_key(C++ member)`,
 107
`_lv_indev_proc_t::last_obj(C++ member)`,
 107
`_lv_indev_proc_t::last_point(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::last_pressed(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::last_raw_point(C++
 member)`, 107
`_lv_indev_proc_t::last_state(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::long_pr_sent(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::longpr_rep_timestamp
 (C++ member)`, 107
`_lv_indev_proc_t::pointer(C++ member)`,
 107
`_lv_indev_proc_t::pr_timestamp(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::reset_query(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::scroll_area(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::scroll_dir(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::scroll_obj(C++ mem-
 ber)`, 107
`_lv_indev_proc_t::scroll_sum(C++ mem-
 ber)`, 107

`_lv_indev_proc_t::scroll_throw_vect (C++ member), 107`
`_lv_indev_proc_t::scroll_throw_vect_ori (C++ member), 107`
`_lv_indev_proc_t::state (C++ member), 107`
`_lv_indev_proc_t::types (C++ member), 107`
`_lv_indev_proc_t::vect (C++ member), 107`
`_lv_indev_proc_t::wait_until_release (C++ member), 107`
`_lv_indev_read (C++ function), 105`
`_lv_indev_t (C++ struct), 108`
`_lv_indev_t::btn_points (C++ member), 108`
`_lv_indev_t::cursor (C++ member), 108`
`_lv_indev_t::driver (C++ member), 108`
`_lv_indev_t::group (C++ member), 108`
`_lv_indev_t::proc (C++ member), 108`
`_lv_obj_t (C++ struct), 291`
`_lv_obj_t::class_p (C++ member), 291`
`_lv_obj_t::coords (C++ member), 291`
`_lv_obj_t::flags (C++ member), 291`
`_lv_obj_t::h_layout (C++ member), 291`
`_lv_obj_t::layout_inv (C++ member), 291`
`_lv_obj_t::parent (C++ member), 291`
`_lv_obj_t::scr_layout_inv (C++ member), 291`
`_lv_obj_t::skip_trans (C++ member), 291`
`_lv_obj_t::spec_attr (C++ member), 291`
`_lv_obj_t::state (C++ member), 291`
`_lv_obj_t::style_cnt (C++ member), 291`
`_lv_obj_t::styles (C++ member), 291`
`_lv_obj_t::user_data (C++ member), 291`
`_lv_obj_t::w_layout (C++ member), 291`
`_lv_style_get_prop_group (C++ function), 150`
`_lv_style_transiton_t (C++ struct), 150`
`_lv_style_transiton_t::delay (C++ member), 151`
`_lv_style_transiton_t::path_xcb (C++ member), 151`
`_lv_style_transiton_t::props (C++ member), 151`
`_lv_style_transiton_t::time (C++ member), 151`
`_lv_style_transiton_t::user_data (C++ member), 151`
`_lv_theme_t (C++ struct), 152`
`_lv_theme_t::apply_cb (C++ member), 153`
`_lv_theme_t::color_primary (C++ member), 153`
`_lv_theme_t::color_secondary (C++ member), 153`
`_lv_theme_t::disp (C++ member), 153`
`_lv_theme_t::flags (C++ member), 153`
`_lv_theme_t::font_large (C++ member), 153`
`_lv_theme_t::font_normal (C++ member), 153`
`_lv_theme_t::font_small (C++ member), 153`
`_lv_theme_t::parent (C++ member), 153`
`_lv_theme_t::user_data (C++ member), 153`
`_lv_timer_core_init (C++ function), 265`
`_lv_timer_t (C++ struct), 267`
`_lv_timer_t::last_run (C++ member), 267`
`_lv_timer_t::paused (C++ member), 267`
`_lv_timer_t::period (C++ member), 267`
`_lv_timer_t::repeat_count (C++ member), 267`
`_lv_timer_t::timer_cb (C++ member), 267`
`_lv_timer_t::user_data (C++ member), 267`
`[anonymous] (C++ enum), 144, 145, 198, 213, 238, 248, 284–286, 297, 308, 324, 371, 383, 392, 403, 415, 432, 441, 448, 468`
`[anonymous]::LV_ARC_MODE_NORMAL (C++ enumerator), 297`
`[anonymous]::LV_ARC_MODE_REVERSE (C++ enumerator), 297`
`[anonymous]::LV_ARC_MODE_SYMMETRICAL (C++ enumerator), 297`
`[anonymous]::LV_BAR_MODE_NORMAL (C++ enumerator), 308`
`[anonymous]::LV_BAR_MODE_RANGE (C++ enumerator), 308`
`[anonymous]::LV_BAR_MODE_SYMMETRICAL (C++ enumerator), 308`
`[anonymous]::LV_BLEND_MODE_ADDITIVE (C++ enumerator), 144`
`[anonymous]::LV_BLEND_MODE_NORMAL (C++ enumerator), 144`
`[anonymous]::LV_BLEND_MODE_SUBTRACTIVE (C++ enumerator), 144`
`[anonymous]::LV_BORDER_SIDE_BOTTOM (C++ enumerator), 145`
`[anonymous]::LV_BORDER_SIDE_FULL (C++ enumerator), 145`
`[anonymous]::LV_BORDER_SIDE_INTERNAL (C++ enumerator), 145`
`[anonymous]::LV_BORDER_SIDE_LEFT (C++ enumerator), 145`
`[anonymous]::LV_BORDER_SIDE_NONE (C++ enumerator), 145`
`[anonymous]::LV_BORDER_SIDE_RIGHT (C++ enumerator), 145`
`[anonymous]::LV_BORDER_SIDE_TOP (C++ enumerator), 145`
`[anonymous]::LV_BTNMATRIX_CTRL_CHECKABLE (C++ enumerator), 324`
`[anonymous]::LV_BTNMATRIX_CTRL_CHECKED (C++ enumerator), 324`
`[anonymous]::LV_BTNMATRIX_CTRL_CLICK_TRIG (C++ enumerator), 324`
`[anonymous]::LV_BTNMATRIX_CTRL_CUSTOM_1`

(C++ enumerator), 325
[anonymous]::LV_BTNMATRIX_CTRL_CUSTOM_2 [anonymous]::LV_FS_RES_OK (C++ enumerator),
(C++ enumerator), 325
[anonymous]::LV_BTNMATRIX_CTRL_DISABLED [anonymous]::LV_FS_RES_OUT_OF_MEM (C++
enumerator), 248
[anonymous]::LV_BTNMATRIX_CTRL_HIDDEN [anonymous]::LV_FS_RES_TOUT (C++ enumera-
tor), 248
[anonymous]::LV_BTNMATRIX_CTRL_NO_REPEAT [anonymous]::LV_FS_RES_UNKNOWN (C++ enu-
merator), 248
[anonymous]::LV_BTNMATRIX_CTRL_RECOLOR [anonymous]::LV_FS_SEEK_CUR (C++ enumera-
tor), 248
[anonymous]::LV_CHART_AXIS_PRIMARY_Y [anonymous]::LV_FS_SEEK_END (C++ enumera-
tor), 248
[anonymous]::LV_CHART_AXIS_SECONDARY_Y [anonymous]::LV_FS_SEEK_SET (C++ enumera-
tor), 248
[anonymous]::LV_CHART_AXIS_X (C++ enumera-
tor), 433
[anonymous]::LV_CHART_TYPE_BAR (C++ enumera-
tor), 432
[anonymous]::LV_CHART_TYPE_LINE (C++ enumera-
tor), 432
[anonymous]::LV_CHART_TYPE_NONE (C++ enumera-
tor), 432
[anonymous]::LV_CHART_UPDATE_MODE_CIRCULAR [anonymous]::LV_GROUP_REFocus_POLICY_NEXT
(C++ enumerator), 432 [anonymous]::LV_GROUP_REFocus_POLICY_PREV
(C++ enumerator), 198
[anonymous]::LV_CHART_UPDATE_MODE_SHIFT [anonymous]::LV_IMG_CF_ALPHA_1BIT (C++
enumerator), 238
[anonymous]::LV_COLORWHEEL_MODE_HUE [anonymous]::LV_IMG_CF_ALPHA_2BIT (C++
enumerator), 238
[anonymous]::LV_COLORWHEEL_MODE_SATURATION [anonymous]::LV_IMG_CF_ALPHA_4BIT (C++
enumerator), 238
[anonymous]::LV_COLORWHEEL_MODE_VALUE [anonymous]::LV_IMG_CF_ALPHA_8BIT (C++
enumerator), 239
[anonymous]::LV_FS_MODE_RD (C++ enumera-
tor), 248
[anonymous]::LV_FS_MODE_WR (C++ enumera-
tor), 248
[anonymous]::LV_FS_RES_BUSY (C++ enumera-
tor), 248
[anonymous]::LV_FS_RES_DENIED (C++ enumera-
tor), 248
[anonymous]::LV_FS_RES_ERR (C++ enumera-
tor), 248
[anonymous]::LV_FS_RES_FULL (C++ enumera-
tor), 248
[anonymous]::LV_FS_RES_HW_ERR (C++ enumera-
tor), 248
[anonymous]::LV_FS_RES_INV_PARAM (C++ enumera-
tor), 248
[anonymous]::LV_FS_RES_LOCKED (C++ enumera-
tor), 248
[anonymous]::LV_FS_RES_NOT_EX (C++ enumera-
tor), 248
[anonymous]::LV_FS_RES_NOT_IMP (C++ enumera-
tor), 248

enumerator), 239

[anonymous]::LV_IMG_CF_RESERVED_19 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_RESERVED_20 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_RESERVED_21 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_RESERVED_22 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_RESERVED_23 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_TRUE_COLOR (C++ *enumerator), 238*

[anonymous]::LV_IMG_CF_TRUE_COLOR_ALPHA (C++ *enumerator), 238*

[anonymous]::LV_IMG_CF_TRUE_COLOR_CHROMA (C++ *enumerator), 238*

[anonymous]::LV_IMG_CF_UNKNOWN (C++ *enumerator), 238*

[anonymous]::LV_IMG_CF_USER_ENCODED_0 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_USER_ENCODED_1 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_USER_ENCODED_2 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_USER_ENCODED_3 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_USER_ENCODED_4 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_USER_ENCODED_5 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_USER_ENCODED_6 (C++ *enumerator), 239*

[anonymous]::LV_IMG_CF_USER_ENCODED_7 (C++ *enumerator), 240*

[anonymous]::LV_KEYBOARD_MODE_NUMBER (C++ *enumerator), 448*

[anonymous]::LV_KEYBOARD_MODE_SPECIAL (C++ *enumerator), 448*

[anonymous]::LV_KEYBOARD_MODE_TEXT_LOWER (C++ *enumerator), 448*

[anonymous]::LV_KEYBOARD_MODE_TEXT_UPPER (C++ *enumerator), 448*

[anonymous]::LV_KEY_BACKSPACE (C++ *enumerator), 198*

[anonymous]::LV_KEY_DEL (C++ *enumerator), 198*

[anonymous]::LV_KEY_DOWN (C++ *enumerator), 198*

[anonymous]::LV_KEY_END (C++ *enumerator), 198*

[anonymous]::LV_KEY_ENTER (C++ *enumerator), 198*

[anonymous]::LV_KEY_ESC (C++ *enumerator), 198*

[anonymous]::LV_KEY_HOME (C++ *enumerator), 198*

[anonymous]::LV_KEY_LEFT (C++ *enumerator), 198*

[anonymous]::LV_KEY_NEXT (C++ *enumerator), 198*

[anonymous]::LV_KEY_PREV (C++ *enumerator), 198*

[anonymous]::LV_KEY_RIGHT (C++ *enumerator), 198*

[anonymous]::LV_KEY_UP (C++ *enumerator), 198*

[anonymous]::LV_LABEL_LONG_CLIP (C++ *enumerator), 371*

[anonymous]::LV_LABEL_LONG_DOT (C++ *enumerator), 371*

[anonymous]::LV_LABEL_LONG_SCROLL (C++ *enumerator), 371*

[anonymous]::LV_LABEL_LONG_SCROLL_CIRCULAR (C++ *enumerator), 371*

[anonymous]::LV_LABEL_LONG_WRAP (C++ *enumerator), 371*

[anonymous]::LV_OBJ_FLAG_ADV_HITTEST (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_CHECKABLE (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_CLICKABLE (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_CLICK_FOCUSABLE (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_EVENT_BUBBLE (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_FLOATING (C++ *enumerator), 287*

[anonymous]::LV_OBJ_FLAG_GESTURE_BUBBLE (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_HIDDEN (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_IGNORE_LAYOUT (C++ *enumerator), 287*

[anonymous]::LV_OBJ_FLAG_LAYOUT_1 (C++ *enumerator), 287*

[anonymous]::LV_OBJ_FLAG_LAYOUT_2 (C++ *enumerator), 287*

[anonymous]::LV_OBJ_FLAG_PRESS_LOCK (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_SCROLLABLE (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_SCROLL_CHAIN (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_SCROLL_ELASTIC (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_SCROLL_MOMENTUM (C++ *enumerator), 286*

[anonymous]::LV_OBJ_FLAG_SCROLL_ONE (C++ enumerator), 286

[anonymous]::LV_OBJ_FLAG_SCROLL_ON_FOCUS (C++ enumerator), 286

[anonymous]::LV_OBJ_FLAG_SNAPABLE (C++ enumerator), 286

[anonymous]::LV_OBJ_FLAG_USER_1 (C++ enumerator), 287

[anonymous]::LV_OBJ_FLAG_USER_2 (C++ enumerator), 287

[anonymous]::LV_OBJ_FLAG_USER_3 (C++ enumerator), 287

[anonymous]::LV_OBJ_FLAG_USER_4 (C++ enumerator), 287

[anonymous]::LV_OBJ_FLAG_WIDGET_1 (C++ enumerator), 287

[anonymous]::LV_OBJ_FLAG_WIDGET_2 (C++ enumerator), 287

[anonymous]::LV_OPA_0 (C++ enumerator), 213

[anonymous]::LV_OPA_10 (C++ enumerator), 213

[anonymous]::LV_OPA_100 (C++ enumerator), 213

[anonymous]::LV_OPA_20 (C++ enumerator), 213

[anonymous]::LV_OPA_30 (C++ enumerator), 213

[anonymous]::LV_OPA_40 (C++ enumerator), 213

[anonymous]::LV_OPA_50 (C++ enumerator), 213

[anonymous]::LV_OPA_60 (C++ enumerator), 213

[anonymous]::LV_OPA_70 (C++ enumerator), 213

[anonymous]::LV_OPA_80 (C++ enumerator), 213

[anonymous]::LV_OPA_90 (C++ enumerator), 213

[anonymous]::LV_OPA_COVER (C++ enumerator), 213

[anonymous]::LV_OPA_TRANSPIRE (C++ enumerator), 213

[anonymous]::LV_PART_ANY (C++ enumerator), 285

[anonymous]::LV_PART_CURSOR (C++ enumerator), 285

[anonymous]::LV_PART_CUSTOM_FIRST (C++ enumerator), 285

[anonymous]::LV_PART_INDICATOR (C++ enumerator), 285

[anonymous]::LV_PART_ITEMS (C++ enumerator), 285

[anonymous]::LV_PART_KNOB (C++ enumerator), 285

[anonymous]::LV_PART_MAIN (C++ enumerator), 285

[anonymous]::LV_PART_SCROLLBAR (C++ enumerator), 285

[anonymous]::LV_PART_SELECTED (C++ enumerator), 285

[anonymous]::LV_PART_TEXTAREA_PLACEHOLDER (C++ enumerator), 415

[anonymous]::LV_PART_TICKS (C++ enumerator), 285

[anonymous]::LV_ROLLER_MODE_INFINITE (C++ enumerator), 383

[anonymous]::LV_ROLLER_MODE_NORMAL (C++ enumerator), 383

[anonymous]::LV_SLIDER_MODE_NORMAL (C++ enumerator), 392

[anonymous]::LV_SLIDER_MODE_RANGE (C++ enumerator), 392

[anonymous]::LV_SLIDER_MODE_SYMMETRICAL (C++ enumerator), 392

[anonymous]::LV_SPAN_MODE_BREAK (C++ enumerator), 468

[anonymous]::LV_SPAN_MODE_EXPAND (C++ enumerator), 468

[anonymous]::LV_SPAN_MODE_FIXED (C++ enumerator), 468

[anonymous]::LV_SPAN_OVERFLOW_CLIP (C++ enumerator), 468

[anonymous]::LV_SPAN_OVERFLOW_ELLIPSIS (C++ enumerator), 468

[anonymous]::LV_STATE_ANY (C++ enumerator), 285

[anonymous]::LV_STATE_CHECKED (C++ enumerator), 284

[anonymous]::LV_STATE_DEFAULT (C++ enumerator), 284

[anonymous]::LV_STATE_DISABLED (C++ enumerator), 285

[anonymous]::LV_STATE_EDITED (C++ enumerator), 285

[anonymous]::LV_STATE_FOCUSED (C++ enumerator), 284

[anonymous]::LV_STATE_FOCUS_KEY (C++ enumerator), 284

[anonymous]::LV_STATE_HOVERED (C++ enumerator), 285

[anonymous]::LV_STATE_PRESSED (C++ enumerator), 285

[anonymous]::LV_STATE_SCROLLED (C++ enumerator), 285

[anonymous]::LV_STATE_USER_1 (C++ enumerator), 285

[anonymous]::LV_STATE_USER_2 (C++ enumerator), 285

[anonymous]::LV_STATE_USER_3 (C++ enumerator), 285

[anonymous]::LV_STATE_USER_4 (C++ enumerator), 285

[anonymous]::LV_TABLE_CELL_CTRL_CUSTOM_1 (C++ enumerator), 403

[anonymous]::LV_TABLE_CELL_CTRL_CUSTOM_2 (C++ enumerator), 403

[anonymous]::LV_TABLE_CELL_CTRL_CUSTOM_3lv_anim_set_playback_delay (C++ function),
 (C++ enumerator), 403
 259

[anonymous]::LV_TABLE_CELL_CTRL_CUSTOM_4lv_anim_set_playback_time (C++ function),
 (C++ enumerator), 403
 258

[anonymous]::LV_TABLE_CELL_CTRL_MERGE_RIGHThanim_set_ready_cb (C++ function), 258
 (C++ enumerator), 403
 lv_anim_set_repeat_count (C++ function), 259

[anonymous]::LV_TABLE_CELL_CTRL_TEXT_CROPrv_anim_set_repeat_delay (C++ function), 259
 (C++ enumerator), 403
 lv_anim_set_start_cb (C++ function), 258

[anonymous]::LV_TEXT_DECOR_NONE (C++ enumerator), 145
 (C++ function), 257
 lv_anim_set_time (C++ function), 257
 lv_anim_set_values (C++ function), 258

[anonymous]::LV_TEXT_DECOR_STRIKETHROUGHlv_anim_set_var (C++ function), 257
 (C++ enumerator), 145
 lv_anim_speed_to_time (C++ function), 260

[anonymous]::LV_TEXT_DECOR_UNDERLINE (C++ enumerator), 145
 (C++ function), 259
 lv_anim_start (C++ function), 259
 lv_anim_start_cb_t (C++ type), 256

[anonymous]::_LV_BTNMATRIX_CTRL_RESERVEDlv_anim_t (C++ type), 257
 (C++ enumerator), 325
 lv_arc_class (C++ member), 300

[anonymous]::_LV_BTNMATRIX_WIDTH (C++ enumerator), 324
 (C++ function), 298
 lv_arc_create (C++ function), 298
 lv_arc_get_angle_end (C++ function), 299

[anonymous]::_LV_CHART_AXIS_LAST (C++ enumerator), 433
 (C++ function), 299
 lv_arc_get_angle_start (C++ function), 299
 lv_arc_get_bg_angle_end (C++ function), 299
 lv_arc_get_bg_angle_start (C++ function), 299

L

lv_anim_count_running (C++ function), 260
 lv_anim_custom_del (C++ function), 260
 lv_anim_custom_exec_cb_t (C++ type), 256
 lv_anim_del (C++ function), 259
 lv_anim_del_all (C++ function), 259
 lv_anim_enable_t (C++ enum), 257
 lv_anim_enable_t::LV_ANIM_OFF (C++ enumerator), 257
 lv_anim_enable_t::LV_ANIM_ON (C++ enumerator), 257
 lv_anim_exec_xcb_t (C++ type), 256
 lv_anim_get (C++ function), 260
 lv_anim_get_delay (C++ function), 259
 lv_anim_get_value_cb_t (C++ type), 256
 lv_anim_init (C++ function), 257
 lv_anim_path_bounce (C++ function), 261
 lv_anim_path_cb_t (C++ type), 256
 lv_anim_path_ease_in (C++ function), 260
 lv_anim_path_ease_in_out (C++ function), 261
 lv_anim_path_ease_out (C++ function), 260
 lv_anim_path_linear (C++ function), 260
 lv_anim_path_overshoot (C++ function), 261
 lv_anim_path_step (C++ function), 261
 lv_anim_ready_cb_t (C++ type), 256
 lv_anim_refr_now (C++ function), 260
 lv_anim_set_custom_exec_cb (C++ function), 258
 lv_anim_set_delay (C++ function), 257
 lv_anim_set_early_apply (C++ function), 259
 lv_anim_set_exec_cb (C++ function), 257
 lv_anim_set_get_value_cb (C++ function), 258
 lv_anim_set_path_cb (C++ function), 258

lv_anim_get_max_value (C++ function), 300
 lv_anim_get_min_value (C++ function), 300
 lv_anim_get_mode (C++ function), 300
 lv_anim_get_value (C++ function), 300
 lv_arc_mode_t (C++ type), 297
 lv_arc_set_angles (C++ function), 298
 lv_arc_set_bg_angles (C++ function), 298
 lv_arc_set_bg_end_angle (C++ function), 298
 lv_arc_set_bg_start_angle (C++ function), 298

lv_arc_set_change_rate (C++ function), 299
 lv_arc_set_end_angle (C++ function), 298
 lv_arc_set_mode (C++ function), 299
 lv_arc_set_range (C++ function), 299
 lv_arc_set_rotation (C++ function), 298
 lv_arc_set_start_angle (C++ function), 298
 lv_arc_set_value (C++ function), 299
 lv_arc_t (C++ struct), 300
 lv_arc_t::bg_angle_end (C++ member), 300
 lv_arc_t::bg_angle_start (C++ member), 300
 lv_arc_t::chg_rate (C++ member), 301
 lv_arc_t::dragging (C++ member), 300
 lv_arc_t::indic_angle_end (C++ member), 300
 lv_arc_t::indic_angle_start (C++ member), 300
 lv_arc_t::last_angle (C++ member), 301
 lv_arc_t::last_tick (C++ member), 301
 lv_arc_t::max_value (C++ member), 300
 lv_arc_t::min_close (C++ member), 301
 lv_arc_t::min_value (C++ member), 300
 lv_arc_t::obj (C++ member), 300

lv_arc_t::rotation (*C++ member*), 300
 lv_arc_t::type (*C++ member*), 300
 lv_arc_t::value (*C++ member*), 300
 lv_async_call (*C++ function*), 267
 lv_async_cb_t (*C++ type*), 267
 lv_bar_anim_t (*C++ struct*), 310
 lv_bar_anim_t::anim_end (*C++ member*), 310
 lv_bar_anim_t::anim_start (*C++ member*),
 310
 lv_bar_anim_t::anim_state (*C++ member*),
 310
 lv_bar_anim_t::bar (*C++ member*), 310
 lv_bar_class (*C++ member*), 310
 lv_bar_create (*C++ function*), 308
 lv_bar_get_max_value (*C++ function*), 309
 lv_bar_get_min_value (*C++ function*), 309
 lv_bar_get_mode (*C++ function*), 309
 lv_bar_get_start_value (*C++ function*), 309
 lv_bar_get_value (*C++ function*), 309
 lv_bar_mode_t (*C++ type*), 308
 lv_bar_set_mode (*C++ function*), 309
 lv_bar_set_range (*C++ function*), 309
 lv_bar_set_start_value (*C++ function*), 308
 lv_bar_set_value (*C++ function*), 308
 lv_bar_t (*C++ struct*), 310
 lv_bar_t::cur_value (*C++ member*), 310
 lv_bar_t::cur_value_anim (*C++ member*), 310
 lv_bar_t::indic_area (*C++ member*), 310
 lv_bar_t::max_value (*C++ member*), 310
 lv_bar_t::min_value (*C++ member*), 310
 lv_bar_t::mode (*C++ member*), 310
 lv_bar_t::obj (*C++ member*), 310
 lv_bar_t::start_value (*C++ member*), 310
 lv_bar_t::start_value_anim (*C++ member*),
 310
 lv_blend_mode_t (*C++ type*), 144
 lv_border_side_t (*C++ type*), 144
 lv_btn_class (*C++ member*), 316
 lv_btn_create (*C++ function*), 316
 lv_btn_t (*C++ struct*), 316
 lv_btn_t::obj (*C++ member*), 316
 lv_btndmatrix_btn_draw_cb_t (*C++ type*), 324
 lv_btndmatrix_class (*C++ member*), 327
 lv_btndmatrix_clear_btn_ctrl (*C++ function*),
 326
 lv_btndmatrix_clear_btn_ctrl_all (*C++ function*), 326
 lv_btndmatrix_create (*C++ function*), 325
 lv_btndmatrix_ctrl_t (*C++ type*), 324
 lv_btndmatrix_get_btn_text (*C++ function*),
 327
 lv_btndmatrix_get_map (*C++ function*), 326
 lv_btndmatrix_get_one_checked (*C++ function*), 327
 lv_btndmatrix_get_selected_btn (*C++ function*), 327
 lv_btndmatrix_has_btn_ctrl (*C++ function*),
 327
 lv_btndmatrix_set_btn_ctrl (*C++ function*),
 325
 lv_btndmatrix_set_btn_ctrl_all (*C++ function*), 326
 lv_btndmatrix_set_btn_width (*C++ function*),
 326
 lv_btndmatrix_set_ctrl_map (*C++ function*),
 325
 lv_btndmatrix_set_map (*C++ function*), 325
 lv_btndmatrix_set_one_checked (*C++ function*), 326
 lv_btndmatrix_set_selected_btn (*C++ function*), 325
 lv_btndmatrix_t (*C++ struct*), 327
 lv_btndmatrix_t::btn_cnt (*C++ member*), 327
 lv_btndmatrix_t::btn_id_sel (*C++ member*),
 327
 lv_btndmatrix_t::button_areas (*C++ member*), 327
 lv_btndmatrix_t::ctrl_bits (*C++ member*),
 327
 lv_btndmatrix_t::map_p (*C++ member*), 327
 lv_btndmatrix_t::obj (*C++ member*), 327
 lv_btndmatrix_t::one_check (*C++ member*),
 327
 lv_calendar_class (*C++ member*), 426
 lv_calendar_create (*C++ function*), 424
 lv_calendar_date_t (*C++ struct*), 426
 lv_calendar_date_t::day (*C++ member*), 426
 lv_calendar_date_t::month (*C++ member*),
 426
 lv_calendar_date_t::year (*C++ member*), 426
 lv_calendar_get_highlighted_dates (*C++ function*), 425
 lv_calendar_get_highlighted_dates_num (*C++ function*), 425
 lv_calendar_get_pressed_date (*C++ function*), 425
 lv_calendar_get_showed_date (*C++ function*),
 425
 lv_calendar_get_today_date (*C++ function*),
 425
 lv_calendar_set_day_names (*C++ function*),
 425
 lv_calendar_set_highlighted_dates (*C++ function*), 425
 lv_calendar_set_showed_date (*C++ function*),
 424
 lv_calendar_set_today_date (*C++ function*),
 424

lv_calendar_t (C++ struct), 426
 lv_calendar_t::btm (C++ member), 426
 lv_calendar_t::highlighted_dates (C++ member), 426
 lv_calendar_t::highlighted_dates_num (C++ member), 426
 lv_calendar_t::map (C++ member), 426
 lv_calendar_t::nums (C++ member), 426
 lv_calendar_t::showed_date (C++ member), 426
 lv_calendar_t::today (C++ member), 426
 lv_canvas_blur_hor (C++ function), 337
 lv_canvas_blur_ver (C++ function), 337
 lv_canvas_copy_buf (C++ function), 336
 lv_canvas_create (C++ function), 335
 lv_canvas_draw_arc (C++ function), 338
 lv_canvas_draw_img (C++ function), 338
 lv_canvas_draw_line (C++ function), 338
 lv_canvas_draw_polygon (C++ function), 338
 lv_canvas_draw_rect (C++ function), 337
 lv_canvas_draw_text (C++ function), 337
 lv_canvas_fill_bg (C++ function), 337
 lv_canvas_get_img (C++ function), 336
 lv_canvas_get_px (C++ function), 336
 lv_canvas_set_buffer (C++ function), 335
 lv_canvas_set_palette (C++ function), 335
 lv_canvas_set_px (C++ function), 335
 lv_canvas_t (C++ struct), 339
 lv_canvas_t::dsc (C++ member), 339
 lv_canvas_t::img (C++ member), 339
 lv_canvas_transform (C++ function), 336
 lv_chart_add_cursor (C++ function), 436
 lv_chart_add_series (C++ function), 435
 lv_chart_axis_t (C++ type), 432
 lv_chart_class (C++ member), 438
 lv_chart_create (C++ function), 433
 lv_chart_cursor_t (C++ struct), 438
 lv_chart_cursor_t::color (C++ member), 438
 lv_chart_cursor_t::dir (C++ member), 438
 lv_chart_cursor_t::point (C++ member), 438
 lv_chart_get_array (C++ function), 437
 lv_chart_get_cursor_point (C++ function), 436
 lv_chart_get_point_count (C++ function), 434
 lv_chart_get_point_pos_by_id (C++ function), 435
 lv_chart_get_pressed_point (C++ function), 437
 lv_chart_get_series_next (C++ function), 436
 lv_chart_get_type (C++ function), 434
 lv_chart_get_x_start_point (C++ function), 435
 lv_chart_get_zoom_x (C++ function), 434
 lv_chart_get_zoom_y (C++ function), 434
 lv_chart_hide_series (C++ function), 435
 lv_chart_refresh (C++ function), 435
 lv_chart_remove_series (C++ function), 435
 lv_chart_series_t (C++ struct), 438
 lv_chart_series_t::color (C++ member), 438
 lv_chart_series_t::ext_buf_assigned (C++ member), 438
 lv_chart_series_t::hidden (C++ member), 438
 lv_chart_series_t::last_point (C++ member), 438
 lv_chart_series_t::points (C++ member), 438
 lv_chart_series_t::y_axis (C++ member), 438
 lv_chart_set_all_value (C++ function), 436
 lv_chart_set_axis_tick (C++ function), 434
 lv_chart_set_cursor_point (C++ function), 436
 lv_chart_set_div_line_count (C++ function), 433
 lv_chart_set_ext_array (C++ function), 437
 lv_chart_set_next_value (C++ function), 437
 lv_chart_set_point_count (C++ function), 433
 lv_chart_set_range (C++ function), 433
 lv_chart_set_series_color (C++ function), 435
 lv_chart_set_type (C++ function), 433
 lv_chart_set_update_mode (C++ function), 433
 lv_chart_set_value_by_id (C++ function), 437
 lv_chart_set_x_start_point (C++ function), 436
 lv_chart_set_zoom_x (C++ function), 434
 lv_chart_set_zoom_y (C++ function), 434
 lv_chart_t (C++ struct), 438
 lv_chart_t::cursor_ll (C++ member), 439
 lv_chart_t::hdiv_cnt (C++ member), 439
 lv_chart_t::obj (C++ member), 439
 lv_chart_t::point_cnt (C++ member), 439
 lv_chart_t::pressed_point_id (C++ member), 439
 lv_chart_t::series_ll (C++ member), 439
 lv_chart_t::tick (C++ member), 439
 lv_chart_t::type (C++ member), 439
 lv_chart_t::update_mode (C++ member), 439
 lv_chart_t::vdiv_cnt (C++ member), 439
 lv_chart_t::ymax (C++ member), 439
 lv_chart_t::ymin (C++ member), 439
 lv_chart_t::zoom_x (C++ member), 439
 lv_chart_t::zoom_y (C++ member), 439
 lv_chart_tick_dsc_t (C++ struct), 438
 lv_chart_tick_dsc_t::draw_size (C++ member), 438

lv_chart_tick_dsc_t::label_en (C++ member), 438
 lv_chart_tick_dsc_t::major_cnt (C++ member), 438
 lv_chart_tick_dsc_t::major_len (C++ member), 438
 lv_chart_tick_dsc_t::minor_cnt (C++ member), 438
 lv_chart_tick_dsc_t::minor_len (C++ member), 438
 lv_chart_type_t (C++ type), 432
 lv_chart_update_mode_t (C++ type), 432
 lv_checkbox_class (C++ member), 343
 lv_checkbox_create (C++ function), 342
 lv_checkbox_get_text (C++ function), 342
 lv_checkbox_set_text (C++ function), 342
 lv_checkbox_set_text_static (C++ function), 342
 lv_checkbox_t (C++ struct), 343
 lv_checkbox_t::obj (C++ member), 343
 lv_checkbox_t::static_txt (C++ member), 343
 lv_checkbox_t::txt (C++ member), 343
 lv_color16_t (C++ union), 216
 lv_color16_t::blue (C++ member), 216
 lv_color16_t::ch (C++ member), 216
 lv_color16_t::full (C++ member), 216
 lv_color16_t::green (C++ member), 216
 lv_color16_t::green_h (C++ member), 216
 lv_color16_t::green_l (C++ member), 216
 lv_color16_t::red (C++ member), 216
 lv_color1_t (C++ union), 215
 lv_color1_t::blue (C++ member), 215
 lv_color1_t::ch (C++ member), 215
 lv_color1_t::full (C++ member), 215
 lv_color1_t::green (C++ member), 215
 lv_color1_t::red (C++ member), 215
 lv_color32_t (C++ union), 216
 lv_color32_t::alpha (C++ member), 216
 lv_color32_t::blue (C++ member), 216
 lv_color32_t::ch (C++ member), 216
 lv_color32_t::full (C++ member), 216
 lv_color32_t::green (C++ member), 216
 lv_color32_t::red (C++ member), 216
 lv_color8_t (C++ union), 215
 lv_color8_t::blue (C++ member), 216
 lv_color8_t::ch (C++ member), 216
 lv_color8_t::full (C++ member), 216
 lv_color8_t::green (C++ member), 216
 lv_color8_t::red (C++ member), 216
 lv_color_black (C++ function), 215
 lv_color_brightness (C++ function), 214
 lv_color_change_lightness (C++ function), 214
 lv_color_chroma_key (C++ function), 215
 lv_color_darken (C++ function), 214
 lv_color_filter_cb_t (C++ type), 212
 lv_color_filter_dsc_init (C++ function), 214
 lv_color_filter_dsc_t (C++ type), 212
 lv_color_hex (C++ function), 214
 lv_color_hex3 (C++ function), 214
 lv_color_hsv_t (C++ struct), 216
 lv_color_hsv_t::h (C++ member), 216
 lv_color_hsv_t::s (C++ member), 216
 lv_color_hsv_t::v (C++ member), 216
 lv_color_hsv_to_rgb (C++ function), 214
 lv_color_lighten (C++ function), 214
 lv_color_make (C++ function), 214
 lv_color_rgb_to_hsv (C++ function), 215
 lv_color_tol (C++ function), 214
 lv_color_to16 (C++ function), 214
 lv_color_to32 (C++ function), 214
 lv_color_to8 (C++ function), 214
 lv_color_to_hsv (C++ function), 215
 lv_color_white (C++ function), 215
 lv_colorwheel_class (C++ member), 442
 lv_colorwheel_create (C++ function), 441
 lv_colorwheel_get_color_mode (C++ function), 442
 lv_colorwheel_get_color_mode_fixed (C++ function), 442
 lv_colorwheel_get_hsv (C++ function), 442
 lv_colorwheel_get_rgb (C++ function), 442
 lv_colorwheel_mode_t (C++ type), 441
 lv_colorwheel_set_hsv (C++ function), 441
 lv_colorwheel_set_mode (C++ function), 441
 lv_colorwheel_set_mode_fixed (C++ function), 442
 lv_colorwheel_set_rgb (C++ function), 441
 lv_colorwheel_t (C++ struct), 442
 lv_colorwheel_t::hsv (C++ member), 442
 lv_colorwheel_t::knob (C++ member), 442
 lv_colorwheel_t::last_change_time (C++ member), 442
 lv_colorwheel_t::last_click_time (C++ member), 442
 lv_colorwheel_t::last_press_point (C++ member), 442
 lv_colorwheel_t::mode (C++ member), 442
 lv_colorwheel_t::mode_fixed (C++ member), 443
 lv_colorwheel_t::obj (C++ member), 442
 lv_colorwheel_t::pos (C++ member), 442
 lv_colorwheel_t::recolor (C++ member), 442
 lv_deinit (C++ function), 287
 lv_disp_clean_dcach (C++ function), 207
 lv_disp_dpx (C++ function), 208
 lv_disp_draw_buf_init (C++ function), 93

lv_disp_draw_buf_t (*C++ struct*), 95
 lv_disp_draw_buf_t::area (*C++ member*), 95
 lv_disp_draw_buf_t::buf1 (*C++ member*), 95
 lv_disp_draw_buf_t::buf2 (*C++ member*), 95
 lv_disp_draw_buf_t::buf_act (*C++ member*),
 95
 lv_disp_draw_buf_t::flushing (*C++ member*), 95
 lv_disp_draw_buf_t::flushing_last (*C++ member*), 95
 lv_disp_draw_buf_t::last_area (*C++ member*), 95
 lv_disp_draw_buf_t::last_part (*C++ member*), 95
 lv_disp_draw_buf_t::size (*C++ member*), 95
 lv_disp_drv_init (*C++ function*), 93
 lv_disp_drv_register (*C++ function*), 93
 lv_disp_drv_t (*C++ type*), 93
 lv_disp_drv_update (*C++ function*), 94
 lv_disp_get_antialiasing (*C++ function*), 94
 lv_disp_get_default (*C++ function*), 94
 lv_disp_get_dpi (*C++ function*), 94
 lv_disp_get_draw_buf (*C++ function*), 95
 lv_disp_get_hor_res (*C++ function*), 94
 lv_disp_get_inactive_time (*C++ function*),
 207
 lv_disp_get_layer_sys (*C++ function*), 206
 lv_disp_get_layer_top (*C++ function*), 206
 lv_disp_get_next (*C++ function*), 95
 lv_disp_get_rotation (*C++ function*), 94
 lv_disp_get_scr_act (*C++ function*), 206
 lv_disp_get_scr_prev (*C++ function*), 206
 lv_disp_get_theme (*C++ function*), 206
 lv_disp_get_ver_res (*C++ function*), 94
 lv_disp_load_scr (*C++ function*), 206
 lv_disp_remove (*C++ function*), 94
 lv_disp_rot_t (*C++ enum*), 93
 lv_disp_rot_t::LV_DISP_ROT_180 (*C++ enumerator*), 93
 lv_disp_rot_t::LV_DISP_ROT_270 (*C++ enumerator*), 93
 lv_disp_rot_t::LV_DISP_ROT_90 (*C++ enumerator*), 93
 lv_disp_rot_t::LV_DISP_ROT_NONE (*C++ enumerator*), 93
 lv_disp_set_bg_color (*C++ function*), 206
 lv_disp_set_bg_image (*C++ function*), 207
 lv_disp_set_bg_opa (*C++ function*), 207
 lv_disp_set_default (*C++ function*), 94
 lv_disp_set_rotation (*C++ function*), 94
 lv_disp_set_theme (*C++ function*), 206
 lv_disp_t (*C++ type*), 93
 lv_disp_trig_activity (*C++ function*), 207
 lv_dpx (*C++ function*), 208

lv_dropdown_add_option (*C++ function*), 350
 lv_dropdown_class (*C++ member*), 353
 lv_dropdown_clear_options (*C++ function*),
 351
 lv_dropdown_close (*C++ function*), 352
 lv_dropdown_create (*C++ function*), 350
 lv_dropdown_get_dir (*C++ function*), 352
 lv_dropdown_get_list (*C++ function*), 351
 lv_dropdown_get_option_cnt (*C++ function*),
 352
 lv_dropdown_get_options (*C++ function*), 352
 lv_dropdown_get_selected (*C++ function*), 352
 lv_dropdown_get_selected_highlight (*C++ function*), 352
 lv_dropdown_get_selected_str (*C++ function*), 352
 lv_dropdown_get_symbol (*C++ function*), 352
 lv_dropdown_get_text (*C++ function*), 351
 lv_dropdown_list_t (*C++ struct*), 353
 lv_dropdown_list_t::dropdown (*C++ member*), 354
 lv_dropdown_list_t::obj (*C++ member*), 354
 lv_dropdown_open (*C++ function*), 352
 lv_dropdown_set_dir (*C++ function*), 351
 lv_dropdown_set_options (*C++ function*), 350
 lv_dropdown_set_options_static (*C++ function*), 350
 lv_dropdown_set_selected (*C++ function*), 351
 lv_dropdown_set_selected_highlight (*C++ function*), 351
 lv_dropdown_set_symbol (*C++ function*), 351
 lv_dropdown_set_text (*C++ function*), 350
 lv_dropdown_t (*C++ struct*), 353
 lv_dropdown_t::dir (*C++ member*), 353
 lv_dropdown_t::list (*C++ member*), 353
 lv_dropdown_t::obj (*C++ member*), 353
 lv_dropdown_t::option_cnt (*C++ member*),
 353
 lv_dropdown_t::options (*C++ member*), 353
 lv_dropdown_t::pr_opt_id (*C++ member*), 353
 lv_dropdown_t::sel_opt_id (*C++ member*),
 353
 lv_dropdown_t::sel_opt_id_orig (*C++ member*), 353
 lv_dropdown_t::selected_highlight (*C++ member*), 353
 lv_dropdown_t::static_txt (*C++ member*),
 353
 lv_dropdown_t::symbol (*C++ member*), 353
 lv_dropdown_t::text (*C++ member*), 353
 lv_dropdownlist_class (*C++ member*), 353
 LV_EXPORT_CONST_INT (*C++ function*), 148, 257,
 325, 350, 371, 403, 415, 433, 488, 493
 lv_flex_align_t (*C++ enum*), 487

lv_flex_align_t::LV_FLEX_ALIGN_CENTER
 (C++ enumerator), 487
 lv_flex_align_t::LV_FLEX_ALIGN_END (C++
 enumerator), 487
 lv_flex_align_t::LV_FLEX_ALIGN_SPACE_AROUND
 (C++ enumerator), 487
 lv_flex_align_t::LV_FLEX_ALIGN_SPACE_BETWEEN
 (C++ enumerator), 487
 lv_flex_align_t::LV_FLEX_ALIGN_SPACE_EVENLY
 (C++ enumerator), 487
 lv_flex_align_t::LV_FLEX_ALIGN_START
 (C++ enumerator), 487
 lv_flex_flow_t (C++ enum), 487
 lv_flex_flow_t::LV_FLEX_FLOW_COLUMN
 (C++ enumerator), 487
 lv_flex_flow_t::LV_FLEX_FLOW_COLUMN_REVERSE
 (C++ enumerator), 487
 lv_flex_flow_t::LV_FLEX_FLOW_COLUMN_WRAP
 (C++ enumerator), 487
 lv_flex_flow_t::LV_FLEX_FLOW_COLUMN_WRAP_REVERSE
 (C++ enumerator), 488
 lv_flex_flow_t::LV_FLEX_FLOW_ROW
 (C++ enumerator), 487
 lv_flex_flow_t::LV_FLEX_FLOW_ROW_REVERSE
 (C++ enumerator), 487
 lv_flex_flow_t::LV_FLEX_FLOW_ROW_WRAP
 (C++ enumerator), 487
 lv_flex_flow_t::LV_FLEX_FLOW_ROW_WRAP_REVERSE
 (C++ enumerator), 487
 lv_flex_init (C++ function), 488
 lv_fs_close (C++ function), 249
 lv_fs_dir_close (C++ function), 250
 lv_fs_dir_open (C++ function), 250
 lv_fs_dir_read (C++ function), 250
 lv_fs_dir_t (C++ struct), 252
 lv_fs_dir_t::dir_d (C++ member), 252
 lv_fs_dir_t::drv (C++ member), 252
 lv_fs_drv_init (C++ function), 249
 lv_fs_drv_register (C++ function), 249
 lv_fs_drv_t (C++ type), 247
 lv_fs_file_t (C++ struct), 251
 lv_fs_file_t::drv (C++ member), 252
 lv_fs_file_t::file_d (C++ member), 252
 lv_fs_get_drv (C++ function), 249
 lv_fs_get_ext (C++ function), 251
 lv_fs_get_last (C++ function), 251
 lv_fs_get_letters (C++ function), 250
 lv_fs_is_ready (C++ function), 249
 lv_fs_mode_t (C++ type), 247
 lv_fs_open (C++ function), 249
 lv_fs_read (C++ function), 249
 lv_fs_res_t (C++ type), 247
 lv_fs_seek (C++ function), 250
 lv_fs_tell (C++ function), 250

lv_fs_up (C++ function), 251
 lv_fs whence_t (C++ type), 247
 lv_fs_write (C++ function), 249
 lv_grad_dir_t (C++ type), 144
 lv_grid_align_t (C++ enum), 493
 lv_grid_align_t::LV_GRID_ALIGN_CENTER
 (C++ enumerator), 493
 lv_grid_align_t::LV_GRID_ALIGN_END (C++
 enumerator), 493
 lv_grid_align_t::LV_GRID_ALIGN_SPACE_AROUND
 (C++ enumerator), 493
 lv_grid_align_t::LV_GRID_ALIGN_SPACE_BETWEEN
 (C++ enumerator), 493
 lv_grid_align_t::LV_GRID_ALIGN_SPACE_EVENLY
 (C++ enumerator), 493
 lv_grid_align_t::LV_GRID_ALIGN_START
 (C++ enumerator), 493
 lv_grid_align_t::LV_GRID_ALIGN_STRETCH
 (C++ enumerator), 493
 lv_grid_init (C++ function), 493
 lv_group_add_obj (C++ function), 199
 lv_group_create (C++ function), 199
 lv_group_del (C++ function), 199
 lv_group_focus_cb_t (C++ type), 198
 lv_group_focus_freeze (C++ function), 199
 lv_group_focus_next (C++ function), 199
 lv_group_focus_obj (C++ function), 199
 lv_group_focus_prev (C++ function), 199
 lv_group_get_default (C++ function), 199
 lv_group_get_editing (C++ function), 200
 lv_group_get_focus_cb (C++ function), 200
 lv_group_get_focused (C++ function), 200
 lv_group_get_wrap (C++ function), 201
 lv_group_refocus_policy_t (C++ type), 198
 lv_group_remove_all_objs (C++ function), 199
 lv_group_remove_obj (C++ function), 199
 lv_group_send_data (C++ function), 200
 lv_group_set_default (C++ function), 199
 lv_group_set_editing (C++ function), 200
 lv_group_set_focus_cb (C++ function), 200
 lv_group_set_refocus_policy (C++ function),
 200
 lv_group_set_wrap (C++ function), 200
 lv_group_t (C++ type), 198
 lv_img_buf_alloc (C++ function), 240
 lv_img_buf_free (C++ function), 241
 lv_img_buf_get_img_size (C++ function), 241
 lv_img_buf_get_px_alpha (C++ function), 240
 lv_img_buf_get_px_color (C++ function), 240
 lv_img_buf_set_palette (C++ function), 241
 lv_img_buf_set_px_alpha (C++ function), 241
 lv_img_buf_set_px_color (C++ function), 240
 lv_img_cf_t (C++ type), 238

`lv_img_class` (*C++ member*), 365
`lv_img_create` (*C++ function*), 363
`lv_img_dsc_t` (*C++ struct*), 243
`lv_img_dsc_t::data` (*C++ member*), 243
`lv_img_dsc_t::data_size` (*C++ member*), 243
`lv_img_dsc_t::header` (*C++ member*), 243
`lv_img_get_angle` (*C++ function*), 364
`lv_img_get_antialias` (*C++ function*), 364
`lv_img_get_offset_x` (*C++ function*), 364
`lv_img_get_offset_y` (*C++ function*), 364
`lv_img_get_pivot` (*C++ function*), 364
`lv_img_get_src` (*C++ function*), 364
`lv_img_get_zoom` (*C++ function*), 364
`lv_img_header_t` (*C++ struct*), 242
`lv_img_header_t::always_zero` (*C++ member*), 242, 243
`lv_img_header_t::cf` (*C++ member*), 242, 243
`lv_img_header_t::h` (*C++ member*), 242, 243
`lv_img_header_t::reserved` (*C++ member*), 242, 243
`lv_img_header_t::w` (*C++ member*), 242, 243
`lv_img_set_angle` (*C++ function*), 363
`lv_img_set_antialias` (*C++ function*), 364
`lv_img_set_offset_x` (*C++ function*), 363
`lv_img_set_offset_y` (*C++ function*), 363
`lv_img_set_pivot` (*C++ function*), 363
`lv_img_set_src` (*C++ function*), 363
`lv_img_set_zoom` (*C++ function*), 363
`lv_img_t` (*C++ struct*), 365
`lv_img_t::angle` (*C++ member*), 365
`lv_img_t::antialias` (*C++ member*), 365
`lv_img_t::cf` (*C++ member*), 365
`lv_img_t::h` (*C++ member*), 365
`lv_img_t::obj` (*C++ member*), 365
`lv_img_t::offset` (*C++ member*), 365
`lv_img_t::pivot` (*C++ member*), 365
`lv_img_t::src` (*C++ member*), 365
`lv_img_t::src_type` (*C++ member*), 365
`lv_img_t::w` (*C++ member*), 365
`lv_img_t::zoom` (*C++ member*), 365
`lv_img_transform_dsc_t` (*C++ struct*), 243
`lv_img_transform_dsc_t::angle` (*C++ member*), 243
`lv_img_transform_dsc_t::antialias` (*C++ member*), 243
`lv_img_transform_dsc_t::cf` (*C++ member*), 243
`lv_img_transform_dsc_t::cfg` (*C++ member*), 243
`lv_img_transform_dsc_t::chroma_keyed` (*C++ member*), 244
`lv_img_transform_dsc_t::color` (*C++ member*), 243
`lv_img_transform_dsc_t::cosma` (*C++ member*), 244
`lv_img_transform_dsc_t::has_alpha` (*C++ member*), 244
`lv_img_transform_dsc_t::img_dsc` (*C++ member*), 243
`lv_img_transform_dsc_t::native_color` (*C++ member*), 244
`lv_img_transform_dsc_t::opa` (*C++ member*), 243
`lv_img_transform_dsc_t::pivot_x` (*C++ member*), 243
`lv_img_transform_dsc_t::pivot_x_256` (*C++ member*), 243
`lv_img_transform_dsc_t::pivot_y` (*C++ member*), 243
`lv_img_transform_dsc_t::pivot_y_256` (*C++ member*), 243
`lv_img_transform_dsc_t::px_size` (*C++ member*), 244
`lv_img_transform_dsc_t::pxi` (*C++ member*), 244
`lv_img_transform_dsc_t::res` (*C++ member*), 243
`lv_img_transform_dsc_t::sinma` (*C++ member*), 244
`lv_img_transform_dsc_t::src` (*C++ member*), 243
`lv_img_transform_dsc_t::src_h` (*C++ member*), 243
`lv_img_transform_dsc_t::src_w` (*C++ member*), 243
`lv_img_transform_dsc_t::tmp` (*C++ member*), 244
`lv_img_transform_dsc_t::xs` (*C++ member*), 244
`lv_img_transform_dsc_t::xs_int` (*C++ member*), 244
`lv_img_transform_dsc_t::ys` (*C++ member*), 244
`lv_img_transform_dsc_t::ys_int` (*C++ member*), 244
`lv_img_transform_dsc_t::zoom` (*C++ member*), 243
`lv_img_transform_dsc_t::zoom_inv` (*C++ member*), 244
`lv_imbtn_class` (*C++ member*), 446
`lv_imbtn_create` (*C++ function*), 445
`lv_imbtn_get_src_left` (*C++ function*), 445
`lv_imbtn_get_src_middle` (*C++ function*), 445
`lv_imbtn_get_src_right` (*C++ function*), 445
`lv_imbtn_set_src` (*C++ function*), 445
`lv_imbtn_state_t` (*C++ enum*), 444
`lv_imbtn_state_t::_LV_IMGBTN_STATE_NUM`

lv_imgbtn_state_t (C++ enumerator), 444
 lv_imgbtn_state_t::LV_IMGBTN_STATE_CHECKED (C++ enumerator), 444
 lv_imgbtn_state_t::LV_IMGBTN_STATE_CHECKED_DISABLED (C++ enumerator), 444
 lv_imgbtn_state_t::LV_IMGBTN_STATE_CHECKED_PRESSED (C++ enumerator), 444
 lv_imgbtn_state_t::LV_IMGBTN_STATE_CHECKED_RELEASED (C++ enumerator), 444
 lv_imgbtn_state_t::LV_IMGBTN_STATE_DISABLED (C++ enumerator), 444
 lv_imgbtn_state_t::LV_IMGBTN_STATE_PRESSED (C++ enumerator), 444
 lv_imgbtn_state_t::LV_IMGBTN_STATE_RELEASED (C++ enumerator), 444
 lv_imgbtn_t (C++ struct), 446
 lv_imgbtn_t::act_cf (C++ member), 446
 lv_imgbtn_t::img_src_left (C++ member), 446
 lv_imgbtn_t::img_src_mid (C++ member), 446
 lv_imgbtn_t::img_src_right (C++ member), 446
 lv_imgbtn_t::obj (C++ member), 446
 lv_indev_data_t (C++ struct), 105
 lv_indev_data_t::btn_id (C++ member), 105
 lv_indev_data_t::continue_reading (C++ member), 106
 lv_indev_data_t::enc_diff (C++ member), 105
 lv_indev_data_t::key (C++ member), 105
 lv_indev_data_t::point (C++ member), 105
 lv_indev_data_t::state (C++ member), 105
 lv_indev_drv_init (C++ function), 105
 lv_indev_drv_register (C++ function), 105
 lv_indev_drv_t (C++ type), 104
 lv_indev_drv_update (C++ function), 105
 lv_indev_enable (C++ function), 196
 lv_indev_get_act (C++ function), 196
 lv_indev_get_gesture_dir (C++ function), 197
 lv_indev_get_key (C++ function), 197
 lv_indev_get_next (C++ function), 105
 lv_indev_get_obj_act (C++ function), 197
 lv_indev_get_point (C++ function), 197
 lv_indev_get_read_timer (C++ function), 197
 lv_indev_get_scroll_dir (C++ function), 197
 lv_indev_get_scroll_obj (C++ function), 197
 lv_indev_get_type (C++ function), 196
 lv_indev_get_vect (C++ function), 197
 lv_indev_proc_t (C++ type), 104
 lv_indev_read_timer_cb (C++ function), 196
 lv_indev_reset (C++ function), 196
 lv_indev_reset_long_press (C++ function), 196
 lv_indev_search_obj (C++ function), 197
 lv_indev_set_button_points (C++ function), 196
 lv_indev_set_cursor (C++ function), 196
 lv_indev_disable_group (C++ function), 196
 lv_indev_state_t (C++ enum), 104
 lv_indev_type_t (C++ enum), 104
 lv_indev_type_t::LV_INDEV_TYPE_BUTTON (C++ enumerator), 104
 lv_indev_type_t::LV_INDEV_TYPE_ENCODER (C++ enumerator), 104
 lv_indev_type_t::LV_INDEV_TYPE_KEYPAD (C++ enumerator), 104
 lv_indev_type_t::LV_INDEV_TYPE_NONE (C++ enumerator), 104
 lv_indev_type_t::LV_INDEV_TYPE_POINTER (C++ enumerator), 104
 lv_indev_wait_release (C++ function), 197
 lv_init (C++ function), 287
 lv_key_t (C++ type), 198
 lv_keyboard_class (C++ member), 450
 lv_keyboard_create (C++ function), 449
 lv_keyboard_def_event_cb (C++ function), 449
 lv_keyboard_get_map_array (C++ function), 449
 lv_keyboard_get_mode (C++ function), 449
 lv_keyboard_get_textarea (C++ function), 449
 lv_keyboard_mode_t (C++ type), 448
 lv_keyboard_set_map (C++ function), 449
 lv_keyboard_set_mode (C++ function), 449
 lv_keyboard_set_textarea (C++ function), 449
 lv_keyboard_t (C++ struct), 450
 lv_keyboard_t::btm (C++ member), 450
 lv_keyboard_t::mode (C++ member), 450
 lv_keyboard_t::ta (C++ member), 450
 lv_label_class (C++ member), 374
 lv_label_create (C++ function), 371
 lv_label_cut_text (C++ function), 373
 lv_label_get_letter_on (C++ function), 373
 lv_label_get_letter_pos (C++ function), 373
 lv_label_get_long_mode (C++ function), 372
 lv_label_get_recolor (C++ function), 372
 lv_label_get_text (C++ function), 372
 lv_label_get_text_selection_end (C++ function), 373
 lv_label_get_text_selection_start (C++ function), 373
 lv_label_ins_text (C++ function), 373
 lv_label_is_char_under_pos (C++ function), 373
 lv_label_long_mode_t (C++ type), 371
 lv_label_set_long_mode (C++ function), 372

lv_label_set_recolor (C++ function), 372
 lv_label_set_text (C++ function), 371
 lv_label_set_text_fmt (C++ function), 372
 lv_label_set_text_sel_end (C++ function),
 372
 lv_label_set_text_sel_start (C++ function),
 372
 lv_label_set_text_static (C++ function), 372
 lv_label_t (C++ struct), 374
 lv_label_t::dot (C++ member), 374
 lv_label_t::dot_end (C++ member), 374
 lv_label_t::dot_tmp_alloc (C++ member),
 374
 lv_label_t::expand (C++ member), 374
 lv_label_t::hint (C++ member), 374
 lv_label_t::long_mode (C++ member), 374
 lv_label_t::obj (C++ member), 374
 lv_label_t::offset (C++ member), 374
 lv_label_t::recolor (C++ member), 374
 lv_label_t::sel_end (C++ member), 374
 lv_label_t::sel_start (C++ member), 374
 lv_label_t::static_txt (C++ member), 374
 lv_label_t::text (C++ member), 374
 lv_label_t::tmp (C++ member), 374
 lv_label_t::tmp_ptr (C++ member), 374
 lv_layer_sys (C++ function), 208
 lv_layer_top (C++ function), 208
 LV_LAYOUT_FLEX (C++ member), 489
 LV_LAYOUT_GRID (C++ member), 496
 lv_led_class (C++ member), 452
 lv_led_create (C++ function), 452
 lv_led_get_brightness (C++ function), 452
 lv_led_off (C++ function), 452
 lv_led_on (C++ function), 452
 lv_led_set_brightness (C++ function), 452
 lv_led_set_color (C++ function), 452
 lv_led_t (C++ struct), 452
 lv_led_t::bright (C++ member), 453
 lv_led_t::color (C++ member), 453
 lv_led_t::obj (C++ member), 453
 lv_led_toggle (C++ function), 452
 lv_line_class (C++ member), 378
 lv_line_create (C++ function), 377
 lv_line_get_y_invert (C++ function), 377
 lv_line_set_points (C++ function), 377
 lv_line_set_y_invert (C++ function), 377
 lv_line_t (C++ struct), 378
 lv_line_t::obj (C++ member), 378
 lv_line_t::point_array (C++ member), 378
 lv_line_t::point_num (C++ member), 378
 lv_line_t::y_inv (C++ member), 378
 lv_list_add_btn (C++ function), 454
 lv_list_add_text (C++ function), 454
 lv_list_btn_class (C++ member), 454
 lv_list_class (C++ member), 454
 lv_list_create (C++ function), 454
 lv_list_get_btn_text (C++ function), 454
 lv_list_text_class (C++ member), 454
 lv_meter_add_arc (C++ function), 460
 lv_meter_add_needle_img (C++ function), 459
 lv_meter_add_needle_line (C++ function), 459
 lv_meter_add_scale (C++ function), 458
 lv_meter_add_scale_lines (C++ function), 460
 lv_meter_class (C++ member), 461
 lv_meter_create (C++ function), 458
 lv_meter_indicator_t (C++ struct), 461
 lv_meter_indicator_t::arc (C++ member),
 462
 lv_meter_indicator_t::color (C++ member),
 462
 lv_meter_indicator_t::color_end (C++
 member), 462
 lv_meter_indicator_t::color_start (C++
 member), 462
 lv_meter_indicator_t::end_value (C++
 member), 462
 lv_meter_indicator_t::local_grad (C++
 member), 462
 lv_meter_indicator_t::needle_img (C++
 member), 462
 lv_meter_indicator_t::needle_line (C++
 member), 462
 lv_meter_indicator_t::opa (C++ member),
 462
 lv_meter_indicator_t::pivot (C++ member),
 462
 lv_meter_indicator_t::r_mod (C++ member),
 462
 lv_meter_indicator_t::scale (C++ member),
 462
 lv_meter_indicator_t::scale_lines (C++
 member), 462
 lv_meter_indicator_t::src (C++ member),
 462
 lv_meter_indicator_t::start_value (C++
 member), 462
 lv_meter_indicator_t::type (C++ member),
 462
 lv_meter_indicator_t::type_data (C++
 member), 462
 lv_meter_indicator_t::width (C++ member),
 462
 lv_meter_indicator_t::width_mod (C++
 member), 462
 lv_meter_indicator_type_t (C++ enum), 458
 lv_meter_indicator_type_t::LV_METER_INDICATOR_TYPE_
 (C++ enumerator), 458
 lv_meter_indicator_type_t::LV_METER_INDICATOR_TYPE_

lv_meter_indicator_type_t::LV_METER_INDICATOR_TYPE_EG_NEDIELINE (C++ enumerator), 458
 lv_meter_indicator_type_t::LV_METER_INDICATOR_TYPE_EG_NEDIELINE+ (C++ enumerator), 458
 lv_meter_indicator_type_t::LV_METER_INDICATOR_TYPE_STATE_CINES (C++ enumerator), 458
 lv_meter_scale_t (C++ struct), 461
 lv_meter_scale_t::angle_range (C++ member), 461
 lv_meter_scale_t::label_color (C++ member), 461
 lv_meter_scale_t::label_gap (C++ member), 461
 lv_meter_scale_t::max (C++ member), 461
 lv_meter_scale_t::min (C++ member), 461
 lv_meter_scale_t::r_mod (C++ member), 461
 lv_meter_scale_t::rotation (C++ member), 461
 lv_meter_scale_t::tick_cnt (C++ member), 461
 lv_meter_scale_t::tick_color (C++ member), 461
 lv_meter_scale_t::tick_length (C++ member), 461
 lv_meter_scale_t::tick_major_color (C++ member), 461
 lv_meter_scale_t::tick_major_length (C++ member), 461
 lv_meter_scale_t::tick_major_nth (C++ member), 461
 lv_meter_scale_t::tick_major_width (C++ member), 461
 lv_meter_scale_t::tick_width (C++ member), 461
 lv_meter_set_indicator_end_value (C++ function), 461
 lv_meter_set_indicator_start_value (C++ function), 460
 lv_meter_set_indicator_value (C++ function), 460
 lv_meter_set_scale_major_ticks (C++ function), 458
 lv_meter_set_scale_range (C++ function), 459
 lv_meter_set_scale_ticks (C++ function), 458
 lv_meter_t (C++ struct), 462
 lv_meter_t::indicator_ll (C++ member), 462
 lv_meter_t::obj (C++ member), 462
 lv_meter_t::scale_ll (C++ member), 462
 lv_msgbox_class (C++ member), 465
 lv_msgbox_close (C++ function), 465
 lv_msgbox_create (C++ function), 464
 lv_msgbox_get_active_btn_text (C++ function), 465
 lv_msgbox_get_btns (C++ function), 465
 lv_msgbox_get_close_btn (C++ function), 465
 lv_obj_allocate_spec_attr (C++ function), 289
 lv_obj_check_type (C++ function), 289
 lv_obj_class (C++ member), 290
 lv_obj_clear_flag (C++ function), 287
 lv_obj_clear_state (C++ function), 288
 lv_obj_create (C++ function), 287
 lv_obj_dpx (C++ function), 290
 lv_obj_flag_t (C++ type), 284
 lv_obj_get_base_dir (C++ function), 288
 lv_obj_get_class (C++ function), 289
 lv_obj_get_group (C++ function), 289
 lv_obj_get_state (C++ function), 289
 lv_obj_get_style_align (C++ function), 154
 lv_obj_get_style_anim_speed (C++ function), 154
 lv_obj_get_style_anim_time (C++ function), 153
 lv_obj_get_style_arc_color (C++ function), 157
 lv_obj_get_style_arc_color_filtered (C++ function), 157
 lv_obj_get_style_arc_img_src (C++ function), 157
 lv_obj_get_style_arc_opa (C++ function), 157
 lv_obj_get_style_arc_rounded (C++ function), 157
 lv_obj_get_style_arc_width (C++ function), 157
 lv_obj_get_style_bg_color (C++ function), 154
 lv_obj_get_style_bg_color_filtered (C++ function), 154
 lv_obj_get_style_bg_grad_color (C++ function), 155
 lv_obj_get_style_bg_grad_color_filtered (C++ function), 155
 lv_obj_get_style_bg_grad_dir (C++ function), 155
 lv_obj_get_style_bg_grad_stop (C++ function), 155
 lv_obj_get_style_bg_img_opa (C++ function), 155
 lv_obj_get_style_bg_img_recolor (C++ function), 155
 lv_obj_get_style_bg_img_recolor_filtered (C++ function), 155
 lv_obj_get_style_bg_img_recolor_opa (C++ function), 155
 lv_obj_get_style_bg_img_src (C++ function),

lv_obj_get_style_bg_img_tiled (C++ function), 155
 lv_obj_get_style_bg_main_stop (C++ function), 155
 lv_obj_get_style_bg_opa (C++ function), 155
 lv_obj_get_style_blend_mode (C++ function), 154
 lv_obj_get_style_border_color (C++ function), 155
 lv_obj_get_style_border_color_filtered (C++ function), 155
 lv_obj_get_style_border_opa (C++ function), 155
 lv_obj_get_style_border_post (C++ function), 155
 lv_obj_get_style_border_side (C++ function), 155
 lv_obj_get_style_border_width (C++ function), 155
 lv_obj_get_style_clip_corner (C++ function), 153
 lv_obj_get_style_color_filter_dsc (C++ function), 153
 lv_obj_get_style_color_filter_opa (C++ function), 153
 lv_obj_get_style_flex_cross_place (C++ function), 489
 lv_obj_get_style_flex_flow (C++ function), 489
 lv_obj_get_style_flex_grow (C++ function), 489
 lv_obj_get_style_flex_main_place (C++ function), 489
 lv_obj_get_style_flex_track_place (C++ function), 489
 lv_obj_get_style_grid_cell_column_pos (C++ function), 495
 lv_obj_get_style_grid_cell_column_span (C++ function), 495
 lv_obj_get_style_grid_cell_row_pos (C++ function), 495
 lv_obj_get_style_grid_cell_row_span (C++ function), 495
 lv_obj_get_style_grid_cell_x_align (C++ function), 495
 lv_obj_get_style_grid_cell_y_align (C++ function), 496
 lv_obj_get_style_grid_column_align (C++ function), 495
 lv_obj_get_style_grid_column_dsc_array (C++ function), 495
 lv_obj_get_style_grid_row_align (C++ function), 495
 lv_obj_get_style_grid_row_dsc_array (C++ function), 495
 lv_obj_get_style_height (C++ function), 154
 lv_obj_get_style_img_opa (C++ function), 156
 lv_obj_get_style_img_recolor (C++ function), 156
 lv_obj_get_style_img_recolor_filtered (C++ function), 156
 lv_obj_get_style_img_recolor_opa (C++ function), 156
 lv_obj_get_style_layout (C++ function), 154
 lv_obj_get_style_line_color (C++ function), 157
 lv_obj_get_style_line_color_filtered (C++ function), 157
 lv_obj_get_style_line_dash_gap (C++ function), 157
 lv_obj_get_style_line_dash_width (C++ function), 157
 lv_obj_get_style_line_opa (C++ function), 157
 lv_obj_get_style_line_rounded (C++ function), 157
 lv_obj_get_style_line_width (C++ function), 157
 lv_obj_get_style_max_height (C++ function), 154
 lv_obj_get_style_max_width (C++ function), 154
 lv_obj_get_style_min_height (C++ function), 154
 lv_obj_get_style_min_width (C++ function), 154
 lv_obj_get_style_opa (C++ function), 153
 lv_obj_get_style_outline_color (C++ function), 156
 lv_obj_get_style_outline_color_filtered (C++ function), 156
 lv_obj_get_style_outline_opa (C++ function), 156
 lv_obj_get_style_outline_pad (C++ function), 156
 lv_obj_get_style_outline_width (C++ function), 156
 lv_obj_get_style_pad_bottom (C++ function), 154
 lv_obj_get_style_pad_column (C++ function), 154
 lv_obj_get_style_pad_left (C++ function), 154
 lv_obj_get_style_pad_right (C++ function), 154
 lv_obj_get_style_pad_row (C++ function), 154
 lv_obj_get_style_pad_top (C++ function), 154

lv_obj_get_style_radius (*C++ function*), 153
 lv_obj_get_style_shadow_color (*C++ function*), 156
 lv_obj_get_style_shadow_color_filtered (*C++ function*), 157
 lv_obj_get_style_shadow_ofs_x (*C++ function*), 156
 lv_obj_get_style_shadow_ofs_y (*C++ function*), 156
 lv_obj_get_style_shadow_opa (*C++ function*), 157
 lv_obj_get_style_shadow_spread (*C++ function*), 156
 lv_obj_get_style_shadow_width (*C++ function*), 156
 lv_obj_get_style_text_align (*C++ function*), 156
 lv_obj_get_style_text_color (*C++ function*), 155
 lv_obj_get_style_text_color_filtered (*C++ function*), 155
 lv_obj_get_style_text_decor (*C++ function*), 156
 lv_obj_get_style_text_font (*C++ function*), 156
 lv_obj_get_style_text_letter_space (*C++ function*), 156
 lv_obj_get_style_text_line_space (*C++ function*), 156
 lv_obj_get_style_text_opa (*C++ function*), 156
 lv_obj_get_style_transform_angle (*C++ function*), 153
 lv_obj_get_style_transform_height (*C++ function*), 153
 lv_obj_get_style_transform_width (*C++ function*), 153
 lv_obj_get_style_transform_zoom (*C++ function*), 153
 lv_obj_get_style_transition (*C++ function*), 154
 lv_obj_get_style_translate_x (*C++ function*), 153
 lv_obj_get_style_translate_y (*C++ function*), 153
 lv_obj_get_style_width (*C++ function*), 154
 lv_obj_get_style_x (*C++ function*), 154
 lv_obj_get_style_y (*C++ function*), 154
 lv_obj_get_user_data (*C++ function*), 289
 lv_obj_has_class (*C++ function*), 289
 lv_obj_has_flag (*C++ function*), 288
 lv_obj_has_flag_any (*C++ function*), 288
 lv_obj_has_state (*C++ function*), 289
 lv_obj_is_valid (*C++ function*), 290

lv_obj_set_base_dir (*C++ function*), 288
 lv_obj_set_flex_align (*C++ function*), 488
 lv_obj_set_flex_flow (*C++ function*), 488
 lv_obj_set_flex_grow (*C++ function*), 488
 lv_obj_set_grid_align (*C++ function*), 493
 lv_obj_set_grid_cell (*C++ function*), 493
 lv_obj_set_grid_dsc_array (*C++ function*), 493
 lv_obj_set_style_align (*C++ function*), 159
 lv_obj_set_style_anim_speed (*C++ function*), 158
 lv_obj_set_style_anim_time (*C++ function*), 158
 lv_obj_set_style_arc_color (*C++ function*), 163
 lv_obj_set_style_arc_color_filtered (*C++ function*), 163
 lv_obj_set_style_arc_img_src (*C++ function*), 163
 lv_obj_set_style_arc_opa (*C++ function*), 163
 lv_obj_set_style_arc_rounded (*C++ function*), 163
 lv_obj_set_style_arc_width (*C++ function*), 163
 lv_obj_set_style_bg_color (*C++ function*), 159
 lv_obj_set_style_bg_color_filtered (*C++ function*), 159
 lv_obj_set_style_bg_grad_color (*C++ function*), 159
 lv_obj_set_style_bg_grad_color_filtered (*C++ function*), 159
 lv_obj_set_style_bg_grad_dir (*C++ function*), 160
 lv_obj_set_style_bg_grad_stop (*C++ function*), 160
 lv_obj_set_style_bg_img_opa (*C++ function*), 160
 lv_obj_set_style_bg_img_recolor (*C++ function*), 160
 lv_obj_set_style_bg_img_recolor_filtered (*C++ function*), 160
 lv_obj_set_style_bg_img_recolor_opa (*C++ function*), 160
 lv_obj_set_style_bg_img_src (*C++ function*), 160
 lv_obj_set_style_bg_img_tiled (*C++ function*), 160
 lv_obj_set_style_bg_main_stop (*C++ function*), 160
 lv_obj_set_style_bg_opa (*C++ function*), 159
 lv_obj_set_style_blend_mode (*C++ function*), 158
 lv_obj_set_style_border_color (*C++ function*)

tion), 160
`lv_obj_set_style_border_color_filtered` (C++ function), 160
`lv_obj_set_style_border_opa` (C++ function), 160
`lv_obj_set_style_border_post` (C++ function), 160
`lv_obj_set_style_border_side` (C++ function), 160
`lv_obj_set_style_border_width` (C++ function), 160
`lv_obj_set_style_clip_corner` (C++ function), 157
`lv_obj_set_style_color_filter_dsc` (C++ function), 158
`lv_obj_set_style_color_filter_opa` (C++ function), 158
`lv_obj_set_style_flex_cross_place` (C++ function), 489
`lv_obj_set_style_flex_flow` (C++ function), 488
`lv_obj_set_style_flex_grow` (C++ function), 489
`lv_obj_set_style_flex_main_place` (C++ function), 489
`lv_obj_set_style_flex_track_place` (C++ function), 489
`lv_obj_set_style_grid_cell_column_pos` (C++ function), 495
`lv_obj_set_style_grid_cell_column_span` (C++ function), 495
`lv_obj_set_style_grid_cell_row_pos` (C++ function), 495
`lv_obj_set_style_grid_cell_row_span` (C++ function), 495
`lv_obj_set_style_grid_cell_x_align` (C++ function), 495
`lv_obj_set_style_grid_cell_y_align` (C++ function), 495
`lv_obj_set_style_grid_column_align` (C++ function), 495
`lv_obj_set_style_grid_column_dsc_array` (C++ function), 494
`lv_obj_set_style_grid_row_align` (C++ function), 494
`lv_obj_set_style_grid_row_dsc_array` (C++ function), 494
`lv_obj_set_style_height` (C++ function), 159
`lv_obj_set_style_img_opa` (C++ function), 161
`lv_obj_set_style_img_recolor` (C++ function), 161
`lv_obj_set_style_img_recolor_filtered` (C++ function), 161
`lv_obj_set_style_img_recolor_opa` (C++ function), 161
`lv_obj_set_style_layout` (C++ function), 159
`lv_obj_set_style_line_color` (C++ function), 162
`lv_obj_set_style_line_color_filtered` (C++ function), 162
`lv_obj_set_style_line_dash_gap` (C++ function), 162
`lv_obj_set_style_line_dash_width` (C++ function), 162
`lv_obj_set_style_line_opa` (C++ function), 163
`lv_obj_set_style_line_rounded` (C++ function), 162
`lv_obj_set_style_line_width` (C++ function), 162
`lv_obj_set_style_max_height` (C++ function), 159
`lv_obj_set_style_max_width` (C++ function), 159
`lv_obj_set_style_min_height` (C++ function), 159
`lv_obj_set_style_min_width` (C++ function), 159
`lv_obj_set_style_opa` (C++ function), 158
`lv_obj_set_style_outline_color` (C++ function), 161
`lv_obj_set_style_outline_color_filtered` (C++ function), 161
`lv_obj_set_style_outline_opa` (C++ function), 162
`lv_obj_set_style_outline_pad` (C++ function), 162
`lv_obj_set_style_outline_width` (C++ function), 161
`lv_obj_set_style_pad_bottom` (C++ function), 158
`lv_obj_set_style_pad_column` (C++ function), 159
`lv_obj_set_style_pad_left` (C++ function), 158
`lv_obj_set_style_pad_right` (C++ function), 158
`lv_obj_set_style_pad_row` (C++ function), 159
`lv_obj_set_style_pad_top` (C++ function), 158
`lv_obj_set_style_radius` (C++ function), 157
`lv_obj_set_style_shadow_color` (C++ function), 162
`lv_obj_set_style_shadow_color_filtered` (C++ function), 162
`lv_obj_set_style_shadow_ofs_x` (C++ function), 162
`lv_obj_set_style_shadow_ofs_y` (C++ function), 162

`lv_obj_set_style_shadow_opa (C++ function),
 162`

`lv_obj_set_style_shadow_spread (C++ function),
 162`

`lv_obj_set_style_shadow_width (C++ function),
 162`

`lv_obj_set_style_text_align (C++ function),
 161`

`lv_obj_set_style_text_color (C++ function),
 161`

`lv_obj_set_style_text_color_filtered
 (C++ function), 161`

`lv_obj_set_style_text_decor (C++ function),
 161`

`lv_obj_set_style_text_font (C++ function),
 161`

`lv_obj_set_style_text_letter_space (C++
 function), 161`

`lv_obj_set_style_text_line_space (C++
 function), 161`

`lv_obj_set_style_text_opa (C++ function),
 161`

`lv_obj_set_style_transform_angle (C++
 function), 158`

`lv_obj_set_style_transform_height (C++
 function), 157`

`lv_obj_set_style_transform_width (C++
 function), 157`

`lv_obj_set_style_transform_zoom (C++
 function), 158`

`lv_obj_set_style_transition (C++ function),
 158`

`lv_obj_set_style_translate_x (C++ func-
 tion), 158`

`lv_obj_set_style_translate_y (C++ func-
 tion), 158`

`lv_obj_set_style_width (C++ function), 159`

`lv_obj_set_style_x (C++ function), 159`

`lv_obj_set_style_y (C++ function), 159`

`lv_obj_set_tile (C++ function), 479`

`lv_obj_set_tile_id (C++ function), 479`

`lv_obj_set_user_data (C++ function), 288`

`lv_obj_spec_attr_t (C++ struct), 290`

`lv_obj_spec_attr_t::base_dir (C++ mem-
 ber), 291`

`lv_obj_spec_attr_t::child_cnt (C++ mem-
 ber), 290`

`lv_obj_spec_attr_t::children (C++ mem-
 ber), 290`

`lv_obj_spec_attr_t::event_dsc (C++ mem-
 ber), 290`

`lv_obj_spec_attr_t::event_dsc_cnt (C++
 member), 291`

`lv_obj_spec_attr_t::ext_click_pad (C++
 member), 290`

`lv_obj_spec_attr_t::ext_draw_size (C++
 member), 290`

`lv_obj_spec_attr_t::group_p (C++ member),
 290`

`lv_obj_spec_attr_t::scroll (C++ member),
 290`

`lv_obj_spec_attr_t::scroll_dir (C++ mem-
 ber), 291`

`lv_obj_spec_attr_t::scroll_snap_x (C++
 member), 291`

`lv_obj_spec_attr_t::scroll_snap_y (C++
 member), 291`

`lv_obj_spec_attr_t::scrollbar_mode (C++
 member), 290`

`lv_obj_t (C++ type), 284`

`lv_palette_darken (C++ function), 215`

`lv_palette_lighten (C++ function), 215`

`lv_palette_main (C++ function), 215`

`lv_palette_t (C++ enum), 213`

`lv_palette_t::LV_PALETTE_LAST (C++ enu-
 merator), 214`

`lv_palette_t::LV_PALETTE_AMBER (C++ enu-
 merator), 213`

`lv_palette_t::LV_PALETTE_BLUE (C++ enu-
 merator), 213`

`lv_palette_t::LV_PALETTE_BLUE_GREY (C++
 enumerator), 214`

`lv_palette_t::LV_PALETTE_BROWN (C++ enu-
 merator), 213`

`lv_palette_t::LV_PALETTE_CYAN (C++ enu-
 merator), 213`

`lv_palette_t::LV_PALETTE_DEEP_ORANGE
 (C++ enumerator), 213`

`lv_palette_t::LV_PALETTE_DEEP_PURPLE
 (C++ enumerator), 213`

`lv_palette_t::LV_PALETTE_GREEN (C++ enu-
 merator), 213`

`lv_palette_t::LV_PALETTE_GREY (C++ enu-
 merator), 214`

`lv_palette_t::LV_PALETTE_INDIGO (C++
 enumerator), 213`

`lv_palette_t::LV_PALETTE_LIGHT_BLUE
 (C++ enumerator), 213`

`lv_palette_t::LV_PALETTE_LIGHT_GREEN
 (C++ enumerator), 213`

`lv_palette_t::LV_PALETTE_LIME (C++ enu-
 merator), 213`

`lv_palette_t::LV_PALETTE_NONE (C++ enu-
 merator), 214`

`lv_palette_t::LV_PALETTE_ORANGE (C++
 enumerator), 213`

`lv_palette_t::LV_PALETTE_PINK (C++ enu-
 merator), 213`

lv_palette_t::LV_PALETTE_PURPLE (C++ enumerator), 213
 lv_palette_t::LV_PALETTE_RED (C++ enumerator), 213
 lv_palette_t::LV_PALETTE_TEAL (C++ enumerator), 213
 lv_palette_t::LV_PALETTE_YELLOW (C++ enumerator), 213
 lv_part_t (C++ type), 284
 lv_roller_class (C++ member), 384
 lv_roller_create (C++ function), 383
 lv_roller_get_option_cnt (C++ function), 384
 lv_roller_get_options (C++ function), 384
 lv_roller_get_selected (C++ function), 384
 lv_roller_get_selected_str (C++ function), 384
 lv_roller_mode_t (C++ type), 383
 lv_roller_set_options (C++ function), 383
 lv_roller_set_selected (C++ function), 383
 lv_roller_set_visible_row_count (C++ function), 383
 lv_roller_t (C++ struct), 384
 lv_roller_t::mode (C++ member), 384
 lv_roller_t::moved (C++ member), 384
 lv_roller_t::obj (C++ member), 384
 lv_roller_t::option_cnt (C++ member), 384
 lv_roller_t::sel_opt_id (C++ member), 384
 lv_roller_t::sel_opt_id_ori (C++ member), 384
 lv_scr_act (C++ function), 207
 lv_scr_load (C++ function), 208
 lv_scr_load_anim (C++ function), 207
 lv_scr_load_anim_t (C++ enum), 205
 lv_scr_load_anim_t::LV_SCR_LOAD_ANIM_FADE (C++ enumerator), 206
 lv_scr_load_anim_t::LV_SCR_LOAD_ANIM_MOVE (C++ enumerator), 206
 lv_scr_load_anim_t::LV_SCR_LOAD_ANIM_MOVELEFT (C++ enumerator), 206
 lv_scr_load_anim_t::LV_SCR_LOAD_ANIM_MOVERIGHT (C++ enumerator), 206
 lv_scr_load_anim_t::LV_SCR_LOAD_ANIM_MOVEUP (C++ enumerator), 206
 lv_scr_load_anim_t::LV_SCR_LOAD_ANIM_MOVEDOWN (C++ enumerator), 206
 lv_scr_load_anim_t::LV_SCR_LOAD_ANIM_NONE (C++ enumerator), 205
 lv_scr_load_anim_t::LV_SCR_LOAD_ANIM_OVER (C++ enumerator), 206
 lv_scr_load_anim_t::LV_SCR_LOAD_ANIM_OVERFLOW (C++ enumerator), 205
 lv_scr_load_anim_t::LV_SCR_LOAD_ANIM_OVERFLOW (C++ enumerator), 206
 lv_slider_class (C++ member), 393
 lv_slider_create (C++ function), 392
 lv_slider_get_left_value (C++ function), 393
 lv_slider_get_max_value (C++ function), 393
 lv_slider_get_min_value (C++ function), 393
 lv_slider_get_mode (C++ function), 393
 lv_slider_get_value (C++ function), 393
 lv_slider_is_dragged (C++ function), 393
 lv_slider_mode_t (C++ type), 392
 lv_slider_set_left_value (C++ function), 392
 lv_slider_set_mode (C++ function), 392
 lv_slider_set_range (C++ function), 392
 lv_slider_set_value (C++ function), 392
 lv_slider_t (C++ struct), 393
 lv_slider_t::bar (C++ member), 394
 lv_slider_t::dragging (C++ member), 394
 lv_slider_t::left_knob_area (C++ member), 394
 lv_slider_t::left_knob_focus (C++ member), 394
 lv_slider_t::right_knob_area (C++ member), 394
 lv_slider_t::value_to_set (C++ member), 394
 lv_span_create (C++ function), 468
 lv_span_del (C++ function), 468
 lv_span_get_align (C++ function), 469
 lv_span_get_expand_height (C++ function), 470
 lv_span_get_expand_width (C++ function), 469
 lv_span_get_indent (C++ function), 469
 lv_span_get_max_line_h (C++ function), 469
 lv_span_get_mode (C++ function), 469
 lv_span_get_overflow (C++ function), 469
 lv_span_mode_t (C++ type), 467
 lv_span_overflow_t (C++ type), 467
 lv_B030Mrefr_mode (C++ function), 470
 lv_span_set_align (C++ function), 469
 lv_B030Lfan_set_indent (C++ function), 469
 lv_B030Rfan_set_mode (C++ function), 469
 lv_B030Tfan_set_overflow (C++ function), 469
 lv_B030span_set_text (C++ function), 468
 lv_B030span_set_text_static (C++ function), 468
 lv_span_t (C++ struct), 470
 lv_B030span_t::static_flag (C++ member), 470
 lv_B030span_t::style (C++ member), 470
 lv_B030Mt::txt (C++ member), 470
 lv_spangroup_class (C++ member), 470
 lv_B030spangroup_create (C++ function), 468
 lv_B030spangroup_t (C++ struct), 470
 lv_B030spangroup_t::align (C++ member), 470
 lv_B030spangroup_t::child_ll (C++ member), 470
 lv_B030spangroup_t::indent (C++ member), 470
 lv_B030spangroup_t::mode (C++ member), 470
 lv_B030spangroup_t::obj (C++ member), 470

`lv_spangroup_t::overflow (C++ member)`, 470
`lv_spinbox_class (C++ member)`, 474
`lv_spinbox_create (C++ function)`, 472
`lv_spinbox_decrement (C++ function)`, 473
`lv_spinbox_get_rollover (C++ function)`, 473
`lv_spinbox_get_step (C++ function)`, 473
`lv_spinbox_get_value (C++ function)`, 473
`lv_spinbox_increment (C++ function)`, 473
`lv_spinbox_set_digit_format (C++ function)`, 472
`lv_spinbox_set_range (C++ function)`, 473
`lv_spinbox_set_rollover (C++ function)`, 472
`lv_spinbox_set_step (C++ function)`, 473
`lv_spinbox_set_value (C++ function)`, 472
`lv_spinbox_step_next (C++ function)`, 473
`lv_spinbox_step_prev (C++ function)`, 473
`lv_spinbox_t (C++ struct)`, 474
`lv_spinbox_t::dec_point_pos (C++ member)`, 474
`lv_spinbox_t::digit_count (C++ member)`, 474
`lv_spinbox_t::range_max (C++ member)`, 474
`lv_spinbox_t::range_min (C++ member)`, 474
`lv_spinbox_t::rollover (C++ member)`, 474
`lv_spinbox_t::step (C++ member)`, 474
`lv_spinbox_t::ta (C++ member)`, 474
`lv_spinbox_t::value (C++ member)`, 474
`lv_spinner_create (C++ function)`, 475
`lv_state_t (C++ type)`, 284
`lv_style_const_prop_t (C++ struct)`, 151
`lv_style_const_prop_t::prop (C++ member)`, 151
`lv_style_const_prop_t::value (C++ member)`, 151
`LV_STYLE_FLEX_CROSS_PLACE (C++ member)`, 489
`LV_STYLE_FLEX_FLOW (C++ member)`, 489
`LV_STYLE_FLEX_GROW (C++ member)`, 489
`LV_STYLE_FLEX_MAIN_PLACE (C++ member)`, 489
`LV_STYLE_FLEX_TRACK_PLACE (C++ member)`, 489
`lv_style_get_prop (C++ function)`, 149
`lv_style_get_prop_inlined (C++ function)`, 149
`LV_STYLE_GRID_CELL_COL_POS (C++ member)`, 496
`LV_STYLE_GRID_CELL_COL_SPAN (C++ member)`, 496
`LV_STYLE_GRID_CELL_ROW_POS (C++ member)`, 496
`LV_STYLE_GRID_CELL_ROW_SPAN (C++ member)`, 496
`LV_STYLE_GRID_CELL_X_ALIGN (C++ member)`, 496
`LV_STYLE_GRID_CELL_Y_ALIGN (C++ member)`, 496
`LV_STYLE_GRID_COLUMN_ALIGN (C++ member)`, 496
`LV_STYLE_GRID_COLUMN_DSC_ARRAY (C++ member)`, 496
`LV_STYLE_GRID_ROW_ALIGN (C++ member)`, 496
`LV_STYLE_GRID_ROW_DSC_ARRAY (C++ member)`, 496
`lv_style_init (C++ function)`, 148
`lv_style_is_empty (C++ function)`, 149
`lv_style_prop_get_default (C++ function)`, 149
`lv_style_prop_t (C++ enum)`, 145
`lv_style_prop_t::LV_STYLE_LAST_BUILT_IN_PROP (C++ enumerator)`, 148
`lv_style_prop_t::LV_STYLE_ALIGN (C++ enumerator)`, 146
`lv_style_prop_t::LV_STYLE_ANIM_SPEED (C++ enumerator)`, 146
`lv_style_prop_t::LV_STYLE_ANIM_TIME (C++ enumerator)`, 146
`lv_style_prop_t::LV_STYLE_ARC_COLOR (C++ enumerator)`, 147
`lv_style_prop_t::LV_STYLE_ARC_COLOR_FILTERED (C++ enumerator)`, 148
`lv_style_prop_t::LV_STYLE_ARC_IMG_SRC (C++ enumerator)`, 148
`lv_style_prop_t::LV_STYLE_ARC_OPA (C++ enumerator)`, 148
`lv_style_prop_t::LV_STYLE_ARC_ROUNDED (C++ enumerator)`, 147
`lv_style_prop_t::LV_STYLE_ARC_WIDTH (C++ enumerator)`, 147
`lv_style_prop_t::LV_STYLE_BG_COLOR (C++ enumerator)`, 146
`lv_style_prop_t::LV_STYLE_BG_COLOR_FILTERED (C++ enumerator)`, 146
`lv_style_prop_t::LV_STYLE_BG_GRAD_COLOR (C++ enumerator)`, 146
`lv_style_prop_t::LV_STYLE_BG_GRAD_COLOR_FILTERED (C++ enumerator)`, 146
`lv_style_prop_t::LV_STYLE_BG_GRAD_DIR (C++ enumerator)`, 146
`lv_style_prop_t::LV_STYLE_BG_GRAD_STOP (C++ enumerator)`, 146
`lv_style_prop_t::LV_STYLE_BG_IMG_OPA (C++ enumerator)`, 146
`lv_style_prop_t::LV_STYLE_BG_IMG_RECOLOR (C++ enumerator)`, 147
`lv_style_prop_t::LV_STYLE_BG_IMG_RECOLOR_FILTERED (C++ enumerator)`, 147
`lv_style_prop_t::LV_STYLE_BG_IMG_RECOLOR_OPA (C++ enumerator)`, 147

lv_style_prop_t::LV_STYLE_BG_IMG_SRC (C++ enumerator), 146	lv_style_prop_t::LV_STYLE_MAX_HEIGHT (C++ enumerator), 146
lv_style_prop_t::LV_STYLE_BG_IMG_TILED (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_MAX_WIDTH (C++ enumerator), 146
lv_style_prop_t::LV_STYLE_BG_MAIN_STOP (C++ enumerator), 146	lv_style_prop_t::LV_STYLE_MIN_HEIGHT (C++ enumerator), 146
lv_style_prop_t::LV_STYLE_BG_OPA (C++ enumerator), 146	lv_style_prop_t::LV_STYLE_MIN_WIDTH (C++ enumerator), 146
lv_style_prop_t::LV_STYLE_BLEND_MODE (C++ enumerator), 146	lv_style_prop_t::LV_STYLE_OPA (C++ enumerator), 146
lv_style_prop_t::LV_STYLE_BORDER_COLOR (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_OUTLINE_COLOR (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_BORDER_COLOR_FILTERED (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_OUTLINE_COLOR_FILTERED (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_BORDER_OPA (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_OUTLINE_OPA (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_BORDER_POST (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_OUTLINE_PAD (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_BORDER_SIDE (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_OUTLINE_WIDTH (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_BORDER_WIDTH (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_PAD_BOTTOM (C++ enumerator), 146
lv_style_prop_t::LV_STYLE_CLIP_CORNER (C++ enumerator), 145	lv_style_prop_t::LV_STYLE_PAD_COLUMN (C++ enumerator), 146
lv_style_prop_t::LV_STYLE_COLOR_FILTER_DBL (C++ enumerator), 146	lv_style_prop_t::LV_STYLE_PAD_LEFT (C++ enumerator), 146
lv_style_prop_t::LV_STYLE_COLOR_FILTER_OPA (C++ enumerator), 146	lv_style_prop_t::LV_STYLE_PAD_RIGHT (C++ enumerator), 146
lv_style_prop_t::LV_STYLE_HEIGHT (C++ enumerator), 146	lv_style_prop_t::LV_STYLE_PAD_ROW (C++ enumerator), 146
lv_style_prop_t::LV_STYLE_IMG_OPA (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_PAD_TOP (C++ enumerator), 146
lv_style_prop_t::LV_STYLE_IMG_RECOLOR (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_PROP_ANY (C++ enumerator), 148
lv_style_prop_t::LV_STYLE_IMG_RECOLOR_FILTERED (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_PROP_INV (C++ enumerator), 145
lv_style_prop_t::LV_STYLE_IMG_RECOLOR_OPA (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_RADIUS (C++ enumerator), 145
lv_style_prop_t::LV_STYLE_LAYOUT (C++ enumerator), 146	lv_style_prop_t::LV_STYLE_SHADOW_COLOR (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_LINE_COLOR (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_SHADOW_COLOR_FILTERED (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_LINE_COLOR_FILTERED (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_SHADOW_OF_X (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_LINE_DASH_GAP (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_SHADOW_OF_Y (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_LINE_DASH_WIDTH (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_SHADOW_OPA (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_LINE_OPA (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_SHADOW_SPREAD (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_LINE_ROUNDED (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_SHADOW_WIDTH (C++ enumerator), 147
lv_style_prop_t::LV_STYLE_LINE_WIDTH (C++ enumerator), 147	lv_style_prop_t::LV_STYLE_TEXT_ALIGN (C++ enumerator), 148

lv_style_prop_t::LV_STYLE_TEXT_COLOR
 (C++ enumerator), 148
 lv_style_prop_t::LV_STYLE_TEXT_COLOR_FILTERED
 (C++ enumerator), 148
 lv_style_prop_t::LV_STYLE_TEXT_DECOR
 (C++ enumerator), 148
 lv_style_prop_t::LV_STYLE_TEXT_FONT
 (C++ enumerator), 148
 lv_style_prop_t::LV_STYLE_TEXT_LINE_SPACEx_style_set_bg_img_src (C++ function), 165
 (C++ enumerator), 148
 lv_style_prop_t::LV_STYLE_TEXT_TRANSFORM_ANGLE
 (C++ enumerator), 146
 lv_style_prop_t::LV_STYLE_TRANSFORM_HEIGHT
 (C++ enumerator), 145
 lv_style_prop_t::LV_STYLE_TRANSFORM_WIDTH
 (C++ enumerator), 145
 lv_style_prop_t::LV_STYLE_TRANSFORM_ZOOM
 (C++ enumerator), 146
 lv_style_prop_t::LV_STYLE_TRANSITION
 (C++ enumerator), 146
 lv_style_prop_t::LV_STYLE_TRANSLATE_X
 (C++ enumerator), 145
 lv_style_prop_t::LV_STYLE_TRANSLATE_Y
 (C++ enumerator), 145
 lv_style_prop_t::LV_STYLE_WIDTH
 (C++ enumerator), 146
 lv_style_prop_t::LV_STYLE_X (C++ enumerator), 146
 lv_style_prop_t::LV_STYLE_Y (C++ enumerator), 146
 lv_style_register_prop (C++ function), 148
 lv_style_remove_prop (C++ function), 148
 lv_style_reset (C++ function), 148
 lv_style_set_align (C++ function), 164
 lv_style_set_anim_speed (C++ function), 164
 lv_style_set_anim_time (C++ function), 164
 lv_style_set_arc_color (C++ function), 167
 lv_style_set_arc_color_filtered (C++ function), 167
 lv_style_set_arc_img_src (C++ function), 167
 lv_style_set_arc_opa (C++ function), 167
 lv_style_set_arc_rounded (C++ function), 167
 lv_style_set_arc_width (C++ function), 167
 lv_style_set_bg_color (C++ function), 165
 lv_style_set_bg_color_filtered (C++ function), 165
 lv_style_set_bg_grad_color (C++ function), 165
 lv_style_set_bg_grad_color_filtered (C++ function), 165
 lv_style_set_bg_grad_dir (C++ function), 165
 lv_style_set_bg_grad_stop (C++ function), 165
 lv_style_set_bg_img_opa (C++ function), 165
 lv_style_set_bg_img_recolor (C++ function), 165
 lv_style_set_bg_img_recolor_filtered (C++ function), 165
 lv_style_set_bg_main_stop (C++ function), 165
 lv_style_set_bg_opa (C++ function), 165
 lv_style_set_blend_mode (C++ function), 164
 lv_style_set_border_color (C++ function), 165
 lv_style_set_border_color_filtered (C++ function), 165
 lv_style_set_border_opa (C++ function), 165
 lv_style_set_border_post (C++ function), 165
 lv_style_set_border_side (C++ function), 165
 lv_style_set_border_width (C++ function), 165
 lv_style_set_clip_corner (C++ function), 163
 lv_style_set_color_filter_dsc (C++ function), 163
 lv_style_set_color_filter_opa (C++ function), 164
 lv_style_set_cross_place (C++ function), 488
 lv_style_set_flex_flow (C++ function), 488
 lv_style_set_flex_grow (C++ function), 488
 lv_style_set_flex_main_place (C++ function), 488
 lv_style_set_grid_cell_column_pos (C++ function), 494
 lv_style_set_grid_cell_column_span (C++ function), 494
 lv_style_set_grid_cell_row_pos (C++ function), 494
 lv_style_set_grid_cell_row_span (C++ function), 494
 lv_style_set_grid_cell_x_align (C++ function), 494
 lv_style_set_grid_cell_y_align (C++ function), 494
 lv_style_set_grid_column_align (C++ function), 494
 lv_style_set_grid_column_dsc_array (C++ function), 494

lv_style_set_align(function), 494
 lv_style_set_grid_row_align(C++ function),
 494
 lv_style_set_grid_row_dsc_array (C++
 function), 494
 lv_style_set_height (C++ function), 164
 lv_style_set_img_opa (C++ function), 166
 lv_style_set_img_recolor (C++ function), 166
 lv_style_set_img_recolor_filtered (C++
 function), 166
 lv_style_set_img_recolor_opa (C++ func-
 tion), 166
 lv_style_set_layout (C++ function), 164
 lv_style_set_line_color (C++ function), 167
 lv_style_set_line_color_filtered (C++
 function), 167
 lv_style_set_line_dash_gap (C++ function),
 167
 lv_style_set_line_dash_width (C++ func-
 tion), 167
 lv_style_set_line_opa (C++ function), 167
 lv_style_set_line_rounded (C++ function),
 167
 lv_style_set_line_width (C++ function), 167
 lv_style_set_max_height (C++ function), 164
 lv_style_set_max_width (C++ function), 164
 lv_style_set_min_height (C++ function), 164
 lv_style_set_min_width (C++ function), 164
 lv_style_set_opa (C++ function), 163
 lv_style_set_outline_color (C++ function),
 166
 lv_style_set_outline_color_filtered
 (C++ function), 166
 lv_style_set_outline_opa (C++ function), 166
 lv_style_set_outline_pad (C++ function), 166
 lv_style_set_outline_width (C++ function),
 166
 lv_style_set_pad_all (C++ function), 150
 lv_style_set_pad_bottom (C++ function), 164
 lv_style_set_pad_column (C++ function), 164
 lv_style_set_pad_gap (C++ function), 150
 lv_style_set_pad_hor (C++ function), 150
 lv_style_set_pad_left (C++ function), 164
 lv_style_set_pad_right (C++ function), 164
 lv_style_set_pad_row (C++ function), 164
 lv_style_set_pad_top (C++ function), 164
 lv_style_set_pad_ver (C++ function), 150
 lv_style_set_prop (C++ function), 148
 lv_style_set_radius (C++ function), 163
 lv_style_set_shadow_color (C++ function),
 166
 lv_style_set_shadow_color_filtered (C++
 function), 167
 lv_style_set_shadow_ofs_x (C++ function),
 166
 lv_style_set_shadow_ofs_y (C++ function),
 166
 lv_style_set_shadow_opa (C++ function), 167
 lv_style_set_shadow_spread (C++ function),
 166
 lv_style_set_shadow_width (C++ function),
 166
 lv_style_set_size (C++ function), 150
 lv_style_set_text_align (C++ function), 166
 lv_style_set_text_color (C++ function), 165
 lv_style_set_text_color_filtered (C++
 function), 166
 lv_style_set_text_decor (C++ function), 166
 lv_style_set_text_font (C++ function), 166
 lv_style_set_text_letter_space (C++ func-
 tion), 166
 lv_style_set_text_line_space (C++ func-
 tion), 166
 lv_style_set_text_opa (C++ function), 166
 lv_style_set_transform_angle (C++ func-
 tion), 163
 lv_style_set_transform_height (C++ func-
 tion), 163
 lv_style_set_transform_width (C++ func-
 tion), 163
 lv_style_set_transform_zoom (C++ function),
 163
 lv_style_set_transition (C++ function), 164
 lv_style_set_translate_x (C++ function), 163
 lv_style_set_translate_y (C++ function), 163
 lv_style_set_width (C++ function), 164
 lv_style_set_x (C++ function), 164
 lv_style_set_y (C++ function), 164
 lv_style_t (C++ struct), 151
 lv_style_t::const_props (C++ member), 151
 lv_style_t::has_group (C++ member), 151
 lv_style_t::is_const (C++ member), 151
 lv_style_t::prop1 (C++ member), 151
 lv_style_t::prop_cnt (C++ member), 151
 lv_style_t::sentinel (C++ member), 151
 lv_style_t::v_p (C++ member), 151
 lv_style_t::value1 (C++ member), 151
 lv_style_t::values_and_props (C++ mem-
 ber), 151
 lv_style_transition_dsc_init (C++ func-
 tion), 149
 lv_style_transition_dsc_t (C++ type), 144
 lv_style_value_t (C++ union), 150
 lv_style_value_t::color (C++ member), 150
 lv_style_value_t::num (C++ member), 150
 lv_style_value_t::ptr (C++ member), 150
 lv_switch_class (C++ member), 396

`lv_switch_create (C++ function)`, 396
`lv_switch_t (C++ struct)`, 396
`lv_switch_t::obj (C++ member)`, 396
`lv_table_add_cell_ctrl (C++ function)`, 404
`lv_table_cell_ctrl_t (C++ type)`, 403
`lv_table_class (C++ member)`, 406
`lv_table_clear_cell_ctrl (C++ function)`, 405
`lv_table_create (C++ function)`, 403
`lv_table_get_cell_value (C++ function)`, 405
`lv_table_get_col_cnt (C++ function)`, 405
`lv_table_get_col_width (C++ function)`, 405
`lv_table_get_row_cnt (C++ function)`, 405
`lv_table_get_selected_cell (C++ function)`, 406
`lv_table_has_cell_ctrl (C++ function)`, 405
`lv_table_set_cell_value (C++ function)`, 403
`lv_table_set_cell_value_fmt (C++ function)`, 404
`lv_table_set_col_cnt (C++ function)`, 404
`lv_table_set_col_width (C++ function)`, 404
`lv_table_set_row_cnt (C++ function)`, 404
`lv_table_t (C++ struct)`, 406
`lv_table_t::cell_data (C++ member)`, 406
`lv_table_t::col_act (C++ member)`, 406
`lv_table_t::col_cnt (C++ member)`, 406
`lv_table_t::col_w (C++ member)`, 406
`lv_table_t::obj (C++ member)`, 406
`lv_table_t::row_act (C++ member)`, 406
`lv_table_t::row_cnt (C++ member)`, 406
`lv_table_t::row_h (C++ member)`, 406
`lv_tabview_add_tab (C++ function)`, 477
`lv_tabview_class (C++ member)`, 477
`lv_tabview_create (C++ function)`, 477
`lv_tabview_get_content (C++ function)`, 477
`lv_tabview_get_tab_act (C++ function)`, 477
`lv_tabview_get_tab_btns (C++ function)`, 477
`lv_tabview_set_act (C++ function)`, 477
`lv_tabview_t (C++ struct)`, 477
`lv_tabview_t::map (C++ member)`, 478
`lv_tabview_t::obj (C++ member)`, 478
`lv_tabview_t::tab_cnt (C++ member)`, 478
`lv_tabview_t::tab_cur (C++ member)`, 478
`lv_tabview_t::tab_pos (C++ member)`, 478
`lv_text_decor_t (C++ type)`, 144
`lv_textarea_add_char (C++ function)`, 415
`lv_textarea_add_text (C++ function)`, 415
`lv_textarea_class (C++ member)`, 419
`lv_textarea_clear_selection (C++ function)`, 419
`lv_textarea_create (C++ function)`, 415
`lv_textarea_cursor_down (C++ function)`, 419
`lv_textarea_cursor_left (C++ function)`, 419
`lv_textarea_cursor_right (C++ function)`, 419
`lv_textarea_cursor_up (C++ function)`, 419

`lv_textarea_del_char (C++ function)`, 416
`lv_textarea_del_char_forward (C++ function)`, 416
`lv_textarea_get_accepted_chars (C++ function)`, 418
`lv_textarea_get_cursor_click_pos (C++ function)`, 418
`lv_textarea_get_cursor_pos (C++ function)`, 418
`lv_textarea_get_label (C++ function)`, 418
`lv_textarea_get_max_length (C++ function)`, 418
`lv_textarea_get_one_line (C++ function)`, 418
`lv_textarea_get_password_mode (C++ function)`, 418
`lv_textarea_get_password_show_time (C++ function)`, 418
`lv_textarea_get_placeholder_text (C++ function)`, 417
`lv_textarea_get_text (C++ function)`, 417
`lv_textarea_get_text_selection (C++ function)`, 418
`lv_textarea_set_accepted_chars (C++ function)`, 416
`lv_textarea_set_align (C++ function)`, 417
`lv_textarea_set_cursor_click_pos (C++ function)`, 416
`lv_textarea_set_cursor_pos (C++ function)`, 416
`lv_textarea_set_insert_replace (C++ function)`, 417
`lv_textarea_set_max_length (C++ function)`, 417
`lv_textarea_set_one_line (C++ function)`, 416
`lv_textarea_set_password_mode (C++ function)`, 416
`lv_textarea_set_password_show_time (C++ function)`, 417
`lv_textarea_set_placeholder_text (C++ function)`, 416
`lv_textarea_set_text (C++ function)`, 416
`lv_textarea_set_text_selection (C++ function)`, 417
`lv_textarea_t (C++ struct)`, 419
`lv_textarea_t::accepted_chars (C++ member)`, 419
`lv_textarea_t::area (C++ member)`, 419
`lv_textarea_t::click_pos (C++ member)`, 419
`lv_textarea_t::cursor (C++ member)`, 420
`lv_textarea_t::label (C++ member)`, 419
`lv_textarea_t::max_length (C++ member)`, 419
`lv_textarea_t::obj (C++ member)`, 419
`lv_textarea_t::one_line (C++ member)`, 420

lv_textarea_t::placeholder_txt (C++ member), 419
 lv_textarea_t::pos (C++ member), 419
 lv_textarea_t::pwd_mode (C++ member), 420
 lv_textarea_t::pwd_show_time (C++ member), 419
 lv_textarea_t::pwd_tmp (C++ member), 419
 lv_textarea_t::sel_end (C++ member), 420
 lv_textarea_t::sel_start (C++ member), 420
 lv_textarea_t::show (C++ member), 419
 lv_textarea_t::text_sel_en (C++ member), 420
 lv_textarea_t::text_sel_in_prog (C++ member), 420
 lv_textarea_t::txt_byte_pos (C++ member), 419
 lv_textarea_t::valid_x (C++ member), 419
 lv_textarea_text_is_selected (C++ function), 418
 lv_theme_apply (C++ function), 152
 lv_theme_apply_cb_t (C++ type), 152
 lv_theme_get_color_primary (C++ function), 152
 lv_theme_get_color_secondary (C++ function), 152
 lv_theme_get_font_large (C++ function), 152
 lv_theme_get_font_normal (C++ function), 152
 lv_theme_get_font_small (C++ function), 152
 lv_theme_get_from_obj (C++ function), 152
 lv_theme_set_apply_cb (C++ function), 152
 lv_theme_set_parent (C++ function), 152
 lv_theme_t (C++ type), 152
 lv_tick_elaps (C++ function), 109
 lv_tick_get (C++ function), 109
 lv_tileview_add_tile (C++ function), 479
 lv_tileview_class (C++ member), 480
 lv_tileview_create (C++ function), 479
 lv_tileview_t (C++ struct), 480
 lv_tileview_t::obj (C++ member), 480
 lv_tileview_tile_class (C++ member), 480
 lv_tileview_tile_t (C++ struct), 480
 lv_tileview_tile_t::dir (C++ member), 480
 lv_tileview_tile_t::obj (C++ member), 480
 lv_timer_cb_t (C++ type), 265
 lv_timer_create (C++ function), 265
 lv_timer_create_basic (C++ function), 265
 lv_timer_del (C++ function), 266
 lv_timer_enable (C++ function), 266
 lv_timer_get_idle (C++ function), 266
 lv_timer_get_next (C++ function), 267
 lv_timer_pause (C++ function), 266
 lv_timer_ready (C++ function), 266
 lv_timer_reset (C++ function), 266
 lv_timer_resume (C++ function), 266
 lv_timer_set_cb (C++ function), 266
 lv_timer_set_period (C++ function), 266
 lv_timer_set_repeat_count (C++ function), 266
 lv_timer_t (C++ type), 265
 lv_win_add_btn (C++ function), 481
 lv_win_add_title (C++ function), 481
 lv_win_class (C++ member), 482
 lv_win_create (C++ function), 481
 lv_win_get_content (C++ function), 481
 lv_win_get_header (C++ function), 481
 lv_win_t (C++ struct), 482
 lv_win_t::obj (C++ member), 482