EE-API.lib

V1.0

Generated by Doxygen 1.8.14

Contents

1 Module Index

1.1 Modules

Here is a list of all modules:

2 Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

3 Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

4 Module Documentation

4.1 defines

Macros

- #define BUFFER_LENGTH 100

  Input buffer length.
- #define LOW_PRIORITY 0

  Interrupt priority.
- #define TIM3_PERIOD 1

  TIM3 period.
- #define TIM3_PRESCALE 16000

  TIM3 prescale. Function = Idle line detect. 1/115200/11 = 10.5kHz basefreq = 2∗APB1 (APB1=42MHz) => TIM_↩ CLK=84MHz Frq = 84MHz/1/16000 = 10,5kHz.
- #define TIM3_REP 0

  TIM3 repetitions.
- #define USART2_BAUT 115200

  USART2 baudrate.

4.1.1 Detailed Description

Group of UART global defines.

4.1.2 Macro Definition Documentation

4.1.2.1 TIM3_PERIOD

```
#define TIM3_PERIOD 1
```

TIM3 period.

Function = Idle line detect. 1/115200/11 = 10.5kHz basefreq = 2∗APB1 (APB1=42MHz) => TIM_CLK=84MHz Frq = 84MHz/1/16000 = 10,5kHz

5 Namespace Documentation

5.1 UART Namespace Reference

namespace UART

Classes

- struct UART_T

    Struct UART_T.

Functions

- void delete_UART (void)

    (GLOBAL) Stops the UART and TIM3
- void disable_IDLE_line (void)

    (GLOBAL) Disable idle line detection.
- void init_GPIO (void)

    (Local) Initiate the GPIO for USART2
- void init_IDLE_Line (void)

    (GLOBAL) Initiate the idle line detection for USART2.
- void init_NVIC (void)

    (Local) Initiate the NVIC for the USART2 interrupt.
- void init_RCC (void)

    (Local) Initiate the needed systemclock
- void init_UART2 (void)

    (GLOBAL) Initiate the UART components.
- void init_USART (void)

    (Local) Initiate USART2
- void put_Char (char c)

(Local) Print char using USART2
- int read (char ∗buf)

  (GLOBAL) Read from the USART2
- void stop_Read (void)

  (GLOBAL) Stops the read loop without user input.
- void TIM3_IRQHandler (void)

  (GLOBAL) TIM3 IRQhandler dor idle line detection.
- void USART2_IRQHandler (void)

  (GLOBAL) IT RXNE interrupt handler for USART2
- void write (char ∗text_out)

  (GLOBAL) Write text using UART2

Variables

- UART_T uart

  Struct with the UART variables.

5.1.1 Detailed Description

namespace UART

Reads and sends data to the user using USART2. Hardware used: USART2, TIM3, USART2 TX PA2, USART2 RX PA3

5.1.2 Function Documentation

5.1.2.1 delete_UART()

```
void UART::delete_UART (
            void  )
```

(GLOBAL) Stops the UART and TIM3

Deletes the UART stops the interrupts and the idle line detection by calling UART::disable_IDLE_Line() if its on.

Parameters

void	

Returns

int error

5.1.2.2 disable_IDLE_line()

```
void UART::disable_IDLE_line (
             void  )
```

(GLOBAL) Disable idle line detection.

Disables TIM3 and TIM3 interrupts.

Parameters

void	

Returns

int error

5.1.2.3 init_GPIO()

```
void UART::init_GPIO (
             void  )
```

(Local) Initiate the GPIO for USART2

Initiate the GPIO needed for USART2

(Pins pack 1) TX = PA2 RX = PA3

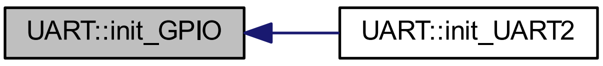
Parameters

void	

Returns

void

Here is the caller graph for this function:



5.1.2.4 init_IDLE_Line()

```
void UART::init_IDLE_Line (
             void  )
```

(GLOBAL) Initiate the idle line detection for USART2.

Setup TIM3 and enable interrupts (TIM3_IRQHandler()).

Parameters

void	

Returns

int error

5.1.2.5 init_NVIC()

```
void UART::init_NVIC (
             void  )
```

(Local) Initiate the NVIC for the USART2 interrupt.

Initiate the NVIC for the UART2 interrupt.

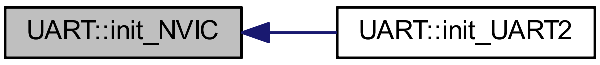
Parameters

void	

Returns

void

Here is the caller graph for this function:



5.1.2.6 init_RCC()

```
void UART::init_RCC (
             void  )
```

(Local) Initiate the needed systemclock

Enables APB1 RCC USART2 Enables APB1 RCC TIM3

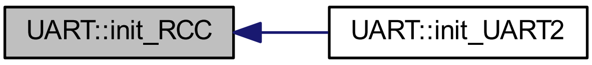
Parameters

void	

Returns

void

Here is the caller graph for this function:



5.1.2.7 init_UART2()

```
void UART::init_UART2 (
             void  )
```

(GLOBAL) Initiate the UART components.

Sets the UART_T struct and initiates the RCC,GPIO,USART and the NVIC

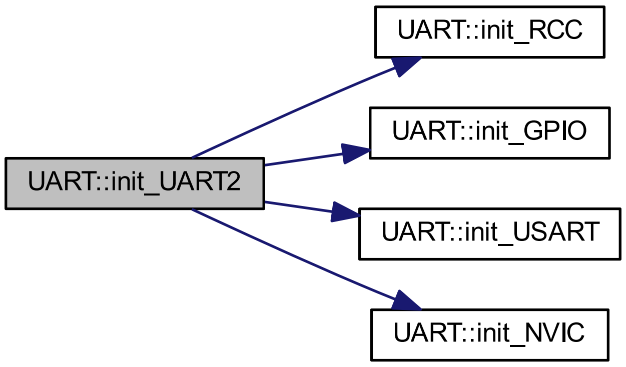
Parameters

void	

Returns

void

Here is the call graph for this function:



5.1.2.8 init_USART()

```
void UART::init_USART (
            void  )
```

(Local) Initiate USART2

- BaudRate = 115200 baud
- Word Length = 8 Bits
- One Stop Bit
- No parity
- Hardware flow control disabled (RTS and CTS signals)
- Receive and transmit enabled

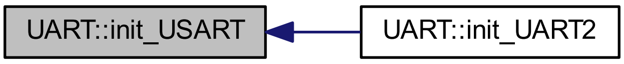
Parameters

void	

Returns

void

Here is the caller graph for this function:



5.1.2.9 put_Char()

```
void UART::put_Char (
             char c )
```

(Local) Print char using USART2

Print a single char.

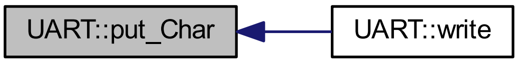
Parameters

char	c, char to print.

Returns

void

Here is the caller graph for this function:



5.1.2.10 read()

```
int UART::read (
             char * buf )
```

(GLOBAL) Read from the USART2

Waits until a input is given or stop_Read() is called.

Parameters

char	∗buf, output buffer with the user input.

Returns

int error

5.1.2.11 stop_Read()

```
void UART::stop_Read (
             void  )
```

(GLOBAL) Stops the read loop without user input.

Stops the read function from waiting for user input.

Parameters

void	

Returns

void

5.1.2.12 TIM3_IRQHandler()

```
void UART::TIM3_IRQHandler (
             void  )
```

(GLOBAL) TIM3 IRQhandler dor idle line detection.

calls the UART::stop_Read() function to continue with executing the buffer.

Parameters

void	

Returns

void

5.1.2.13 USART2_IRQHandler()

```
void UART::USART2_IRQHandler (
             void   )
```

(GLOBAL) IT RXNE interrupt handler for USART2

Fills the input buffer and sets the UART::UART_T.bReady flag when done.

Parameters

void	

Returns

void

5.1.2.14 write()

```
void UART::write (
             char * text_out )
```

(GLOBAL) Write text using UART2

prints a string char by char using the put_Char() function..

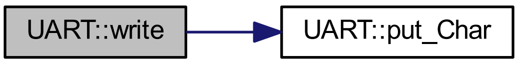
Parameters

char	*text_out, the text that needs to be printed.

Returns

void

Here is the call graph for this function:



5.1.3 Variable Documentation

5.1.3.1 uart

```
UART_T UART::uart
```

Struct with the UART variables.

6 Class Documentation

6.1 UART::UART_T Struct Reference

Struct UART_T.

```
#include "Uart.h"
```

Public Attributes

- volatile int bReady
- volatile int iIndex
- volatile int iLine
- char inputBuffer [BUFFER_LENGTH]
- volatile int rCancel
- volatile int uBusy

6.1.1 Detailed Description

Struct UART_T.

Contains the different variables needed for the UART to work.

6.1.2 Member Data Documentation

6.1.2.1 bReady

```
volatile int UART::UART_T::bReady
```

Buffer ready FLAG. To signal if there's any user input.

6.1.2.2 iIndex

```
volatile int UART::UART_T::iIndex
```

Input index, keeps track what the next position is in the buffer.

6.1.2.3 iLine

```
volatile int UART::UART_T::iLine
```

Idle line FLAG to enable/disable idle line detection.

6.1.2.4 inputBuffer

```
char UART::UART_T::inputBuffer[BUFFER_LENGTH]
```

Input buffer with size BUFFER_LENGTH.

6.1.2.5 rCancel

```
volatile int UART::UART_T::rCancel
```

read cancel FLAG. to cancel the read function.

6.1.2.6 uBusy

```
volatile int UART::UART_T::uBusy
```

USART busy flag, needed for the idle line detection.

6.2 VGA_t Struct Reference

Public Attributes

- uint32_t **dma2_cr_reg**
- uint16_t **hsync_cnt**
- uint32_t **start_adr**

6.3 Vgascreen Class Reference

The VGAscreen class.

```
#include "Vgascreen.h"
```

Public Member Functions

- int clear_screen (int color)

  Fill screen with specified color.
- int draw_bitmap (int nr, int x_lo, int y_lo)

  Draw bitmap on screen.
- int draw_ellipse (int x_mp, int y_mp, int x_rad, int y_rad, int color, int fill)

  Draw ellipse on screen.
- int draw_line (int x1, int y1, int x2, int y2, int width, int color)

  Draw line on screen.
- int draw_rectangle (int x_lo, int y_lo, int x_rb, int y_rb, int color, int fill)

  Draw rectangle on screen.
- int draw_text (int x, int y, const char ∗str, int color, const char ∗style, int fontNr)

  Draw text on screen.
- int draw_triangle (int x1, int y1, int x2, int y2, int x3, int y3, int color, int fill)

  Draw triangle on screen.
- Vgascreen (void)

  Create VGAscreen object.
- virtual ∼Vgascreen (void)

  Destroy the VGAscreen object.

6.3.1 Detailed Description

The VGAscreen class.

The VGAscreen is a class that enables the VGA screen using the UB library. The different methods of class are to draw different shapes on the VGA screen.

can draw: lijn, rechthoek, ellips, driehoek, tekst and bitmaps. (lines, rectangles, ellipse, triangle, text and bitmaps)

6.3.2 Constructor & Destructor Documentation

6.3.2.1 Vgascreen()

```
Vgascreen::Vgascreen (
            void )
```

Create VGAscreen object.

Parameters

void.	

6.3.2.2 ∼Vgascreen()

```
Vgascreen::∼Vgascreen (
            void )  [virtual]
```

Destroy the VGAscreen object.

Parameters

void.	

6.3.3 Member Function Documentation

6.3.3.1 clear_screen()

```
int Vgascreen::clear_screen (
            int color )
```

Fill screen with specified color.

Clear the screen in a specific color.

Parameters

int	color, 256-color value.

Returns

int error

6.3.3.2 draw_bitmap()

```
int Vgascreen::draw_bitmap (
             int nr,
             int x_lo,
             int y_lo )
```

Draw bitmap on screen.

Draws a bitmap from the bootem left coordinate. bitmaps: 0=smileface, 1=sadface, 2=arrow up, 3=arrow down.

Parameters

int	nr, number of the bitmap to be displayed (0=smileface, 1=sadface, 2=arrow up, 3=arrow down).
int	x_lo, bottom left x coordinate.
int	y_lo, bottom left y coordinate.

Returns

int error

6.3.3.3 draw_ellipse()

```
int Vgascreen::draw_ellipse (
             int x_mp,
             int y_mp,
             int x_rad,
             int y_rad,
             int color,
             int fill )
```

Draw ellipse on screen.

Draws an ellipse on the screen using a middle point and the x and y radius

Parameters

int	x_mp, x coordinate of the middle point.
int	y_mp, x coordinate of the middle point.
int	x_rad, radius in the x direction.
int	y_rad, radius in the y direction.
int	color, 256-color value.
int	fill, fill the ellipse.

Returns

int error

6.3.3.4 draw_line()

```
int Vgascreen::draw_line (
             int x1,
             int y1,
             int x2,
             int y2,
             int width,
             int color )
```

Draw line on screen.

Draws a line between 2 points (x1,y1,x2,y2).

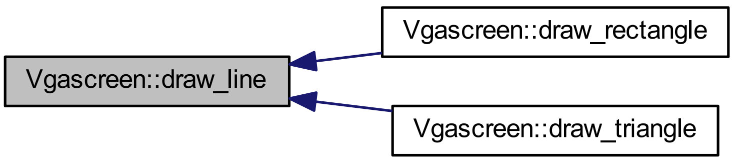
Parameters

int	x1, the first x coordinate of the line.
int	y1, the first y coordinate of the line.
int	x2, the second x coordinate of the line.
int	y2, the second y coordinate of the line.
int	witdh, the witdh of the line.
int	color, 256-color value.

Returns

int error

Here is the caller graph for this function:



6.3.3.5 draw_rectangle()

```
int Vgascreen::draw_rectangle (
              int x_lo,
              int y_lo,
              int x_rb,
              int y_rb,
              int color,
              int fill )
```

Draw rectangle on screen.

Draws a rectangle from the bottom left corner(x_lo,y_lo) to the top right corner (x_rb, y_rb).

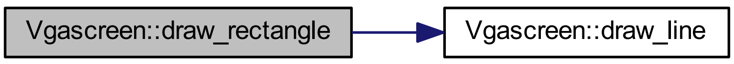
Parameters

int	x_lo, bottom left x coordinate.
int	y_lo, bottom left y coordinate.
int	x_rb, top right x coordinate.
int	y_rb, top right y coordinate.
int	color, 256-color value.
int	fill, fill the rectangle.

Returns

int error

Here is the call graph for this function:



6.3.3.6 draw_text()

```
int Vgascreen::draw_text (
              int x,
              int y,
              const char * str,
              int color,
              const char * style,
              int fontNr )
```

Draw text on screen.

Draws the given text on the screen. there are 2 fonts ("1"/"2") and 3 styles ("norm","cursief","vet").

Parameters

int	x, x coordiante bottom left of the text.
int	y, y coordiante bottom left of the text.
const	char ∗str, Text to be displayed.
int	color, text color (256-color value).
int	style, style of the text ("norm","cursief","vet").
int	fontNr, select differtn fonts ("1"/"2").

Returns

int error

6.3.3.7 draw_triangle()

```
int Vgascreen::draw_triangle (
            int x1,
            int y1,
            int x2,
            int y2,
            int x3,
            int y3,
            int color,
            int fill )
```

Draw triangle on screen.

Draws a triangle between 3 points ((x1,y1),(x2,y2),(x3,y3).

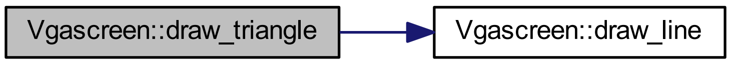
Parameters

int	x1, x coordinate of the first point.
int	y1, y coordinate of the first point.
int	x2, x coordinate of the second point.
int	y2, y coordinate of the second point.
int	x3, x coordinate of the third point.
int	y3, y coordinate of the third point.
int	color, 256-color value.
int	fill, fill the rectangle.

Returns

int error

Here is the call graph for this function:



Index