

vesofon\_otd

Generated by Doxygen 1.9.3



<b>1 VGA Graphics library</b>	<b>1</b>
1.1 Flowchart	1
1.2 Script usages	2
1.3 Adding fonts and bitmaps	2
1.4 Debugging	2
1.5 Error handling	2
<b>2 Script tutorial</b>	<b>3</b>
2.1 Line	3
2.2 Rectangle	4
2.3 Text	4
2.4 Bitmap	5
2.5 Clearscreen	6
2.6 Wait	6
2.7 Repeat	6
2.8 Circle	7
2.9 Figure	7
2.10 Colors	8
<b>3 Adding fonts and bitmap to code</b>	<b>9</b>
3.1 Bitmap	9
3.2 Fonts	9
<b>4 Data Structure Index</b>	<b>11</b>
4.1 Data Structures	11
<b>5 File Index</b>	<b>13</b>
5.1 File List	13
<b>6 Data Structure Documentation</b>	<b>15</b>
6.1 input_vars Struct Reference	15
6.2 wait_vars Struct Reference	15
<b>7 File Documentation</b>	<b>17</b>
7.1 Core/Inc/API.h File Reference	17
7.1.1 Detailed Description	18
7.1.2 Function Documentation	18
7.1.2.1 drawBitmap()	18
7.1.2.2 drawCircle()	19
7.1.2.3 drawCircleplus()	21
7.1.2.4 drawFigure()	21
7.1.2.5 drawLines()	22
7.1.2.6 drawParallelogram()	23
7.1.2.7 drawPixel()	24

7.1.2.8 drawRect()	24
7.1.2.9 drawText()	25
7.1.2.10 myLijntekenaar()	26
7.2 API.h	27
7.3 Core/Inc/bitmap.h File Reference	28
7.3.1 Detailed Description	29
7.3.2 Variable Documentation	29
7.3.2.1 smiley_map	29
7.4 bitmap.h	29
7.5 Core/Inc/error.h File Reference	204
7.5.1 Detailed Description	205
7.6 error.h	205
7.7 Core/Inc/front_layer.h File Reference	206
7.7.1 Detailed Description	206
7.7.2 Function Documentation	206
7.7.2.1 FL_UART_Init()	207
7.8 front_layer.h	207
7.9 Core/Inc/logic_layer.h File Reference	207
7.9.1 Detailed Description	208
7.9.2 Function Documentation	208
7.9.2.1 buffer_init()	209
7.9.2.2 HAL_TIM_PeriodElapsedCallback()	209
7.9.2.3 indexBuffer()	209
7.9.2.4 logic_layer()	210
7.9.2.5 writeBuffer()	210
7.10 logic_layer.h	210
7.11 Core/Inc/main.h File Reference	211
7.11.1 Detailed Description	212
7.11.2 Function Documentation	212
7.11.2.1 Error_Handler()	212
7.12 main.h	213
7.13 Core/Inc/stm32f4xx_it.h File Reference	214
7.13.1 Detailed Description	215
7.14 stm32f4xx_it.h	215
7.15 Core/Src/API.c File Reference	216
7.15.1 Detailed Description	217
7.15.2 Function Documentation	217
7.15.2.1 drawBitmap()	217
7.15.2.2 drawCircle()	218
7.15.2.3 drawCircleplus()	218
7.15.2.4 drawFigure()	219
7.15.2.5 drawLines()	219

7.15.2.6 drawParallelogram()	220
7.15.2.7 drawPixel()	221
7.15.2.8 drawRect()	221
7.15.2.9 drawText()	222
7.15.2.10 myLijntekenaar()	223
7.16 Core/Src/front_layer.c File Reference	224
7.16.1 Detailed Description	224
7.16.2 Function Documentation	225
7.16.2.1 FL_UART_Init()	225
7.17 Core/Src/logic_layer.c File Reference	225
7.17.1 Detailed Description	226
7.17.2 Function Documentation	226
7.17.2.1 buffer_init()	226
7.17.2.2 font_style()	226
7.17.2.3 HAL_TIM_PeriodElapsedCallback()	227
7.17.2.4 indexBuffer()	227
7.17.2.5 logic_layer()	227
7.17.2.6 r3g3b2_Colour()	228
7.17.2.7 writeBuffer()	228
7.18 Core/Src/main.c File Reference	228
7.18.1 Detailed Description	229
7.18.2 Function Documentation	229
7.18.2.1 Error_Handler()	229
7.18.2.2 main()	230
7.18.2.3 SystemClock_Config()	230
7.18.3 Variable Documentation	231
7.18.3.1 USART_PRINTF	231
7.19 Core/Src/stm32f4xx_it.c File Reference	231
7.19.1 Detailed Description	232



# Chapter 1

## VGA Graphics library

With this software you can easily print different types of graphics on a vga screen over UART.

This software can generate a VGA signal of 240 x 320 pixels.

### 1.1 Flowchart

The beautiful flowchart.

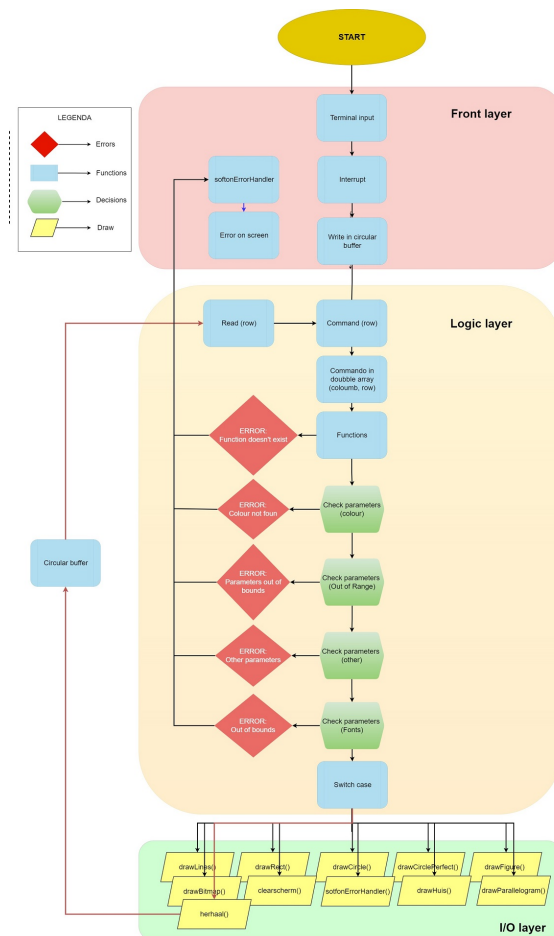


Figure 1.1 Flowchart

## 1.2 Script usages

see [Script](#)

## 1.3 Adding fonts and bitmaps

see [Adding](#)

## 1.4 Debugging

In different header files are some debug defines that are commanded out. By adding those debug defines in the code can be enabled or disabled.

## 1.5 Error handling

If an error occurs the error will be printed over the uart and the error code will be printed over VGA on the screen.



## Chapter 2

# Script tutorial

In this explanation all the scripts will be explained what they can do.

### 2.1 Line

With this script command a line can be printed on the screen.  
The syntax for this script is:

```
lijn, x1, y1, x2, y2, kleur, dikte, middeliijn
```

- **x1:** This is the x of the first coordinate.
- **y1:** This is the y of the first coordinate.
- **x2:** This is the x of the second coordinate.
- **y2:** This is the y of the second coordinate.
- **kleur:** This is the color of the line to be draw. see [Colors](#) for the differnt colors to be used.
- **dikte:** This is the width of the line in pixels.
- **middeliijn:** If middeliijn is equal to one,the function draws a black line with the original coordinates.

## 2.2 Rectangle

With this script command a rectangle can be printed on the screen.  
The syntax for this script is:

```
rechthoek, x_lup, y_lup, breedte, hoogte, kleur, gevuld, rand, randkleur
```

- **x\_up**: This is the x coordinate of the left upper corner.
- **y\_up**: This is the x coordinate of the left upper corner.
- **breedte**: This is the width of the rectangle in pixels.
- **hoogte**: This is the height of the rectangle in pixels.
- **kleur**: This is the color of the line to be draw. see [Colors](#) for the differnt colors to be used.
- **gevuld**: This is for filling the rectangle.
  - 0 for not filled.
  - 1 for filled.
- **rand**: this is if a border around the rectangle will be drone.
  - 0 for no border.
  - 1 for a border
- **randkleur**: this is the border color. see [Colors](#) for the differnt colors to be used.

## 2.3 Text

With this script command a piece of text can be printed on the screen.  
The syntax for this script is:

```
tekst, x, y, kleur, tekst, fontnaam, fontgrootte, fontstijl, space, linefeed
```

- **x**: This is the x coordinate of the left upper corner.
- **y**: This is the x coordinate of the left upper corner.
- **kleur**: This is the color of the line to be draw. see [Colors](#) for the differnt colors to be used.
- **tekst**: This is the text that will be printed on the screen. If a comma needs te be printed a backslash can be put before it to prevent it to be seen als script delimiter. If tekst ends with a backslash be sure to put a space between the backslash and de comma delimiter to prevent it to be printed on the screen.

- **fontnaam:** This is the name of the font to be used. Available are:
  - arial.
  - consolas.
- **fontgrootte:** This is the size of the fonts. Available are:
  - 1 for 8px.
  - 2 for 32px.
- **fontstijl:** This is for filling the rectangle. Available are:
  - "normaal" for normal.
  - "vet" for bold.
  - "cursief" for Italic.
- **space:** This is the space between the characters in pixels.
- **linefeed:** This is if you would like to go to the next line if the characters go out of the screen.
  - 0 for not next line.
  - 1 for next line.

## 2.4 Bitmap

With this script command a bitmap can be printed on the screen.  
The syntax for this script is:

```
bitmap, nr, x-lup, y-lup
```

- **nr:** This is the number of the bitmap to be used. Standard available are:

Number	Bitmap
0	Smiley
1	Angry smiley
2	Arrow left
3	Arrow right
4	Arrow up
5	Arrow down
6	Smiley 2
7	Angry smiley 2
8	Arrow right 2

- **x-1up**: This is the x coordinate of the left upper corner.
- **y-1up**: This is the y coordinate of the left upper corner.

## 2.5 Clearscreen

With this script command the screen can be cleared.  
The syntax for this script is:

```
clearscreen, kleur
```

- **kleur**: This is the color of the line to be draw. see [Colors](#) for the different colors to be used.

## 2.6 Wait

With the script a specific time to wait can be implemented  
The syntax for this script is:

```
wacht, msec
```

- **msec**: Time to wait in milliseconds.

## 2.7 Repeat

With the script a specific number of scripts can be repeated.  
The syntax for this script is:

```
herhaal, aantal, hoevaak, richting
```

- **aantal**: This is the number of scripts to be repeated.
- **hoevaak**: This is how many times the scripts have to be repeated.
- **richting**: This is in which order the scripts should be repeated.
  - 0 for normal order.
  - 1 for turned order.

## 2.8 Circle

With the script a specific time to wait can be implemented  
The syntax for this script is:

```
cirkel, x, y, radius, kleur, lradius
```

- **x:** This is the x coordinate of the middlepoint.
- **y:** This is the y coordinate of the middlepoint.
- **radius:** This is the radius of the circle.
- **kleur:** This is the color of the line to be draw. see [Colors](#) for the differnt colors to be used.
- **lradius:** This is the lower boundary radius of the circle.

## 2.9 Figure

With the script a specific figure can be printed on the screen.  
The syntax for this script is:

```
figuur, x1,y1, x2,y2, x3,y3, x4,y4, x5,y5, kleur, dikte
```

- **x1:** the x of the first coordinate.
- **y1:** the y of the first coordinate.
- **y2:** the y of the second coordinate.
- **x2:** the x of the second coordinate.
- **x3:** the x of the third coordinate.
- **y3:** the y of the third coordinate.
- **x4:** the x of the fourth coordinate.
- **y4:** the y of the fourth coordinate.
- **x5:** the x of the fifth coordinate.
- **y5:** the y of the fifth coordinate.
- **kleur:** This is the color of the line to be draw. see [Colors](#) for the differnt colors to be used.
- **dikte:** This is the width of the line in pixels.

## 2.10 Colors

In this section all the available colors are listed and which string needs to be added to the script.

Color	Script string
Black	zwart
Blue	blauw
Light blue	lichtblauw
Green	groen
Light green	lichtgroen
Red	rood
Light Red	lichtrood
White	wit
Cyan	cyaan
Light cyan	lichtcyaan
Magenta	magenta
Light magenta	lichtmagenta
Brown	bruin
Yellow	geel
Gray	grijs
Pink	roze
Violet	paars

## Chapter 3

# Adding fonts and bitmap to code

On this page we will discuss how different fonts and bitmaps can be added to the code.

### 3.1 Bitmap

To add a bitmap to the code follow the following steps.

1. Add bitmap array into [bitmap.h](#)
2. change the first byte to the height of the picture
3. change the second and third byte to the width of the picture whereby the second byte is the higher byte and the third byte is the lower byte.
4. At the bottom of [bitmap.h](#) change the array and add the name of bitmap.
5. You have successfully added a bitmap to the code.

### 3.2 Fonts

To add a font to the code follow the following steps.

Credit of the script for font generation is going to <https://github.com/jdmorise>.

1. create a font with the python script included in the repository.
  - (a) run the python script in folder Font generator\TTF2BMH-master\src\ttf2bmh.py with the following arguments  

```
python3 .\ttf2bmh.py --ascii --font <font> --variable_width -s <Hight in pixels> --square
```

and generate the font to your likings in normal, Bold and italic in 8px height and 32px height.
2. add the files to /Core/Inc/fonts.
3. In [API.h](#) increase the value of FONT\_NUMBER with 1.
4. In [API.c](#) include all the font files
5. In [API.c](#) in the function [drawText\(\)](#) add to the variable `**fonts[]` the different fonts in the same order as the previous fonts.
6. In [API.c](#) in the function [drawText\(\)](#) add to the variable `font_name` the font name you added to be called by the script.
7. You now have successfully added a font to the code.





## Chapter 4

# Data Structure Index

### 4.1 Data Structures

Here are the data structures with brief descriptions:

<a href="#">input_vars</a>	.....	15
<a href="#">wait_vars</a>	.....	15



## Chapter 5

# File Index

### 5.1 File List

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

Core/Inc/ <a href="#">API.h</a>	
: Library for Graphics . . . . .	17
Core/Inc/ <a href="#">bitmap.h</a>	
: bitmap arrays to be printed on the screen . . . . .	28
Core/Inc/ <a href="#">error.h</a>	
: Defines for all error codes . . . . .	204
Core/Inc/ <a href="#">front_layer.h</a>	206
Core/Inc/ <a href="#">logic_layer.h</a>	
: logic layer . . . . .	207
Core/Inc/ <a href="#">main.h</a>	
: Header for <a href="#">main.c</a> file. This file contains the common defines of the application . . . . .	211
Core/Inc/ <a href="#">stm32f4xx_it.h</a>	
This file contains the headers of the interrupt handlers . . . . .	214
Core/Src/ <a href="#">API.c</a>	
: Library for Graphics . . . . .	216
Core/Src/ <a href="#">front_layer.c</a>	224
Core/Src/ <a href="#">logic_layer.c</a>	
: Logic layer . . . . .	225
Core/Src/ <a href="#">main.c</a>	
: Main program body . . . . .	228
Core/Src/ <a href="#">stm32f4xx_it.c</a>	
Interrupt Service Routines . . . . .	231



## Chapter 6

# Data Structure Documentation

### 6.1 input\_vars Struct Reference

#### Data Fields

- uint8\_t **byte\_buffer\_rx** [BYTE\_BUFLen]
- char **line\_rx\_buffer** [LINE\_BUFLen]
- int **msglen**
- volatile int **char\_counter**
- char **command\_execute\_flag**

The documentation for this struct was generated from the following file:

- Core/Inc/[front\\_layer.h](#)

### 6.2 wait\_vars Struct Reference

#### Data Fields

- char **buffer** [BUFFER\_SIZE][MAX\_NUMBER\_OF\_SCRIPT\_CHARACTER]
- int **counter**
- int **waitFlag**
- int **head**
- int **executed**

The documentation for this struct was generated from the following file:

- Core/Inc/[logic\\_layer.h](#)



## Chapter 7

# File Documentation

### 7.1 Core/Inc/API.h File Reference

: Library for Graphics.

```
#include <stdio.h>
#include <stdint.h>
#include <math.h>
#include <string.h>
#include <stdarg.h>
#include <stdlib.h>
#include "stm32_ub_vga_screen.h"
```

#### Macros

- `#define __weak __attribute__((weak))`
- `#define BEGGINING_OF_BITMAP 3`
- `#define BITMAP_WIDTH_HIGH 2`
- `#define BITMAP_WIDTH_LOW 1`
- `#define BITMAP_LENGTH 0`
- `#define FONT_TYPES 6`
- `#define BYTE_SIZE 8`
- `#define FONT_NUMBER 2`
- `#define FONT_NAME_SIZE 20`
- `#define FONT_SIZE 2`
- `#define FONT_STYLE 2`
- `#define SPACE_BETWEEN_LETTER 2`
- `#define LETTER_BITS 31`
- `#define LETTER_WIDTH_BITS 32`
- `#define FONT_SIZE_POSITION 3`
- `#define COLOURMAX 255`
- `#define COLOURMIN 0`
- `#define LINE_WIDTH 10`
- `#define X_BEGIN 0`
- `#define Y_BEGIN 0`
- `#define FIGURE_POINTS 5`
- `#define RADIUS_INCREMENT_CIRCLE 360`
- `#define HALF_CIRCLE_DEGREE 180`

## Functions

- `__weak void drawPixel (uint16_t xp, uint16_t yp, uint8_t colour)`  
*Weak declared Draw pixel function for if the user wants to implement his own display interface.*
- `int drawBitmap (int nr, uint8_t x_1up, uint8_t y_1up)`  
*this function shows a graphics image file (bitmaps) on VGA display.*
- `int drawText (int x, int y, uint8_t colour, char tekst[], char fontname[], uint8_t fontgrootte, uint8_t fontstyle, uint8_t spaces, uint8_t linefeed)`  
*This function prints the text on the screen.*
- `int myLijntekenaar (uint16_t x_begin, uint16_t y_begin, uint16_t x_eind, uint16_t y_eind, uint8_t colour)`  
*this function draws a single line from a to b*
- `int drawLines (uint16_t x_begin, uint16_t y_begin, uint16_t x_eind, uint16_t y_eind, uint8_t colour, uint8_t lijn_dikte, uint8_t mid_lijn)`  
*This function draws multiple lines using mylijntekenaar function.*
- `int drawRect (uint16_t x_pos, uint16_t y_pos, uint16_t length, uint16_t width, uint8_t colour, uint8_t filled, uint8_t rectborder, uint8_t border_colour)`  
*This function draws a rectangle either filled or not.*
- `int drawCircle (uint16_t x_pos, uint16_t y_pos, uint8_t radius, uint8_t colour, uint8_t lradius)`  
*This function draws a circle.*
- `int drawFigure (uint8_t colour, uint8_t lijn_dikte, uint8_t nr_pointsgiven,...)`  
*This functions draw a figure with X amount of points.*
- `int drawCircleplus (uint16_t x_pos, uint16_t y_pos, uint8_t l_radius, uint8_t h_radius, uint16_t l_angle, uint16_t h_angle, uint8_t colour)`  
*This function draws a circle.*
- `int drawParallelogram (uint16_t x_pos, uint16_t y_pos, uint16_t length, uint16_t width, uint8_t angle, uint8_t colour)`  
*This function draws an parallelogram using length,width and an angle.*

### 7.1.1 Detailed Description

: Library for Graphics.

### 7.1.2 Function Documentation

#### 7.1.2.1 drawBitmap()

```
int drawBitmap (
    int nr,
    uint8_t x_1up,
    uint8_t y_1up )
```

this function shows a graphics image file (bitmaps) on VGA display.

Bitmap, method by which a display space (such as a graphics image file) is defined, including the colour of each of its pixels (or bits). Is an array of binary data representing the values of pixels in an image or display. This function has 3 variables, the number of the bitmap and the position of bitmap. Bitmap has 6 icons; smile, angry and the four arrows. The user can choose between smile=0, angry=1, left=2, up=3, right=4, right=5. Add more bitmap guide: Step 1: Add bitmap array into [bitmap.h](#) Step 2: Add 3 bytes at the position 0, 1 and 2 of array: 2a: Position 0 is the Y boundary in HEX. 2b: Position 1&2 are the X boundary in HEX. Step 3: At the bottom of [bitmap.h](#) add the names of bitmaps into the array list.



## Parameters

<i>nr</i>	is the bitmap number to display.
<i>x_1up</i>	and <i>y_1up</i> is the left upper position of the bitmap

## Return values

<i>Error</i>	
--------------	--

## Author

Djalil & Tjerk

## Note

Bitmap size must be smaller or equal to VGA resolution.

## Warning

this function shows a graphics image file (bitmaps) on VGA display.

Bitmap has 6 icons; smiley, angry and the four arrows. smiley\_map=0, angry\_map=1, Arrow left=2, Arrow up= The user can choose between smiley=0, angry=1, left=2, up=3, right=4, right=5. Add more bitmapguide: Step 1: Add bitmap array into [bitmap.h](#) Step 2: At the bottom of [bitmap.h](#) change the array and add the name of bitmap.

## Parameters

<i>nr</i>	is the bitmap number to draw.
<i>x_1up</i>	and <i>y_1up</i> is the position of the bitmap

## Return values

<i>Error</i>	
--------------	--

## Author

Djalil & Tjerk

### 7.1.2.2 drawCircle()

```
int drawCircle (  
    uint16_t x_pos,  
    uint16_t y_pos,
```

```
uint8_t radius,  
uint8_t colour,  
uint8_t lradius )
```

This function draws a circle.

By using **elementary** mathematics the function draw a circle. The for loop draws points of the circle until it drew the whole circle. The for loop goes through every angle between 0 - 2 PI radius

#### Parameters

<i>x_pos,y_pos</i>	These are the x and y coordinates of the middle of the circle.
<i>radius</i>	This is the radius of the circle.
<i>colour</i>	This is the colour of the circle.
<i>lradius</i>	This is the lower boundary of radius

mathematic proof

#### Return values

Error	
-------	--

#### Author

Osman Pekcan

By using **elementary** mathematics the function draw a circle. The for loop draws points of the circle until it drew the whole circle. The for loop goes through every angle between 0 - 2 PI radius

#### Parameters

<i>x_pos,y_pos</i>	These are the x and y coordinates of the middle of the circle.
<i>radius</i>	This is the radius of the circle.
<i>colour</i>	This is the colour of the circle.
<i>lradius</i>	This is the lower boundary of radius

mathematic proof

#### Return values

Error	
-------	--

#### Author

Osman Pekcan

### 7.1.2.3 drawCircleplus()

```
int drawCircleplus (
    uint16_t x_pos,
    uint16_t y_pos,
    uint8_t l_radius,
    uint8_t h_radius,
    uint16_t l_angle,
    uint16_t h_angle,
    uint8_t colour )
```

This function draws a circle.

By using **elementary** mathematics the function draw a circle. The for loop draws points of the circle until it drew the whole circle. The for loop goes through every angle between 0 - 2 PI radius

#### Parameters

<i>x_pos,y_pos</i>	These are the x and y coordinates of the middle of the circle.
<i>radius</i>	This is the radius of the circle.
<i>angle</i>	the angle where to start and stop the circle
<i>colour</i>	This is the colour of the circle <i>mathematic proof</i>

#### Return values

<i>Error</i>	
--------------	--

#### Author

Osman Pekcan

### 7.1.2.4 drawFigure()

```
int drawFigure (
    uint8_t colour,
    uint8_t lijn_dikte,
    uint8_t nr_pointsgiven,
    ... )
```

This functions draw a figure with X amount of points.

With the function mylijntekenaar,this function draws multiple line that are chained to each other. With *va\_list* the argument after *nr\_pointsgiven* are saved in a list. Using *va\_arg* the argument are put into two array *figure\_ram\_x* and *figure\_ram\_y*. with arithmetic code the argument that represent x coordinate are put into *figure\_ram\_x*. likewise for the y coordinates. with these to arrays the function can draw its figure with a for loop.

#### Parameters

<i>colour</i>	This parameter is for colour of the lines.
<i>lijn_dikte</i>	The variable that dictate how big the line width is.
<i>nr_pointgiven</i>	Locked the number of points

## Return values

<i>Error</i>	
--------------	--

## Author

Osman Pekcan

### 7.1.2.5 drawLines()

```
int drawLines (
    uint16_t x_begin,
    uint16_t y_begin,
    uint16_t x_eind,
    uint16_t y_eind,
    uint8_t colour,
    uint8_t lijn_dikte,
    uint8_t mid_lijn )
```

This function draws multiple lines using mylijntekenaar function.

By repeating the function mylijntekenaar, this draws multiple lines. To do this it checks the parameter which concludes the width of the sum of lines. If it is one, the function just calls the function mylijntekenaar once. Else draws the lines around the middle line(original x and y positions). For the function to know whether to draw extra line in the x axis or y axis it looks at the derivative. If the derivative is higher than one its draw from the x axis, else its draws from the y axis

## Parameters

<i>lijn_dikte</i>	The variable that dictate how big the line width is.
<i>mid_lijn</i>	If mid_lijn is equal to one,the function draws a black line with the original coordinates.

## See also

[myLijntekenaar\(\)](#)

## Return values

<i>Error</i>	
--------------	--

**Author**

Osman Pekcan

Here is the call graph for this function:

**7.1.2.6 drawParallelogram()**

```
int drawParallelogram (
    uint16_t x_pos,
    uint16_t y_pos,
    uint16_t length,
    uint16_t width,
    uint8_t angle,
    uint8_t colour )
```

This function draws an parallelogram using length,width and an angle.

**Parameters**

<i>x_pos</i>	X coordinate of top left corner
<i>y_pos</i>	Y coordinate of top left corner
<i>length</i>	The length of the parallelogram
<i>width</i>	The width of the parallelogram
<i>angle</i>	The angle of the bottom corner
<i>colour</i>	Colour of the parallelogram

**Return values**

<i>Error</i>	
--------------	--

**Author**

Osman Pekcan

### 7.1.2.7 drawPixel()

```
__weak void drawPixel (
    uint16_t xp,
    uint16_t yp,
    uint8_t colour )
```

Weak declared Draw pixel function for if the user wants to implement his own display interface.

By declaring this function in the user his own code the user can implement his own display interface code and still use this graphics library.

#### Parameters

<i>x</i>	x coordinate of the pixel
<i>y</i>	y coordinate of the pixel
<i>color</i>	color of the pixel

#### Author

Tjerk ten Dam

### 7.1.2.8 drawRect()

```
int drawRect (
    uint16_t x_pos,
    uint16_t y_pos,
    uint16_t length,
    uint16_t width,
    uint8_t colour,
    uint8_t filled,
    uint8_t rectborder,
    uint8_t border_colour )
```

This function draws a rectangle either filled or not.

This function draw either a filled or unfilled rectangle, using filled parameter to decide which one. If it filled, its loops from y axis, drawing lines (using mylijntekenaar) from x axis. If it unfilled draws four lines of the unfilled rectangle

#### Parameters

<i>x_pos,y_pos</i>	Upper left position of the rectangle.
<i>length,width</i>	Length and width of the rectangle.
<i>colour</i>	colour of rectangle.
<i>filled</i>	Parameter that determines whether the rectangle is filled or not
<i>rectborder</i>	Parameter that determines if the function draws a border around the filled rectangle.
<i>border_colour</i>	The color value of the border.

## Return values

Error	
-------	--

## Author

Osman Pekcan

## 7.1.2.9 drawText()

```
int drawText (
    int x,
    int y,
    uint8_t colour,
    char tekst[],
    char fontname[],
    uint8_t fontsize,
    uint8_t fontstyle,
    uint8_t spaces,
    uint8_t linefeed )
```

This function prints the text on the screen.

This function prints the text on the screen. The user can choose between Arial= 0 en Consolas=1. Each fonts have 3 styles: Standard, Italic and Bold Each fonts have 2 sizes: 8, 32 Fontstyle Standard=0, Italic=1 and Bold=2. Fontsize 8=0, 32=1. Add more fonts: Step 1: Import the header file into fonts folder. Step 2: Add all the fonts into `**fonts[]`. Step 3: Add fontnames into `font_name[][]`. Step 4: Change the Define `FONT_NUMBER`

## Parameters

<i>x,y</i>	Is the position of the starting point of the first letter.
<i>colour</i>	Is the fonts colour.
<i>tekst[]</i>	Is an array with text.
<i>fontname</i>	is the fonts: Arial and Consolas.
<i>fontsize</i>	is the size of the font. The size can be chosen from 8 or 32. 0=8 and 1=32.
<i>frontstyle</i>	is the style of fonts, this can be chosen of Italic, Bold.

## Author

Djalil & Tjerk

## Return values

Error	
-------	--

## Note

**Warning**

This function prints the text on the screen.

This function prints/draws the text on the screen. The user can choose between Arial= 0 en Consolas=1. Each fonts have 3 styles: Standard, Italic and Bold Each fonts have 2 sizes: 8, 32 Fontstyle Standard=0, Italic=1 and Bold=2. Fontsize 8=0, 32=1. Add more fonts: Step 1: Import the header file into fonts folder. Step 2: Add all the fonts into `**fonts[]`. Step 3: Add fontnames into `font_name[][]`. Step 4: Change the Define FONT\_NUMBER

**Parameters**

<i>x,y</i>	Is the position of the starting point of the first letter.
<i>colour</i>	Is the fonts colour.
<i>tekst[]</i>	Is an array with text.
<i>fontname</i>	Can be chosen from fonts of Arial and Consolas.
<i>fontsize</i>	is the size of the font. The size can be chosen from 8 or 32. 0 = 8 and 1 = 32.
<i>frontstyle</i>	is the style of fonts, this can be chosen of Italic, Bold.

**Return values**

<i>Error</i>	
--------------	--

**Author**

Djalil & Tjerk

**7.1.2.10 myLijntekenaar()**

```
int myLijntekenaar (
    uint16_t x_begin,
    uint16_t y_begin,
    uint16_t x_eind,
    uint16_t y_eind,
    uint8_t colour )
```

this function draws a single line from a to b

This function starts with 5 variables, x and y coordinate for the begin and x and y coordinates for the end. The last variable as colour of the line. The function first calculates the offset of both x and y coordinates. When either offsets is equal to zero, the line becomes straight. If y offset is zero the function draws horizontal line, else a vertical line. In the situation that neither offset are zero, the function calculates the derivative of the line. The derivative of the line determines the angle that the line has. A derivative value lower than one has a small angle. If the angle is *small* (derivative<1) the y coordinates is calculated with the derivative. If the angle is *big* (derivative>1) it is the inverse of the calculation of the smaller angle. After the function looks whether the derivative is higher than one, the function looks if the derivative is positive or negative. This determines if some parameter is positive or negative. Lastly the function looks at orientation of the line with respect to the x axis, which decides which parameter is the start point and end point of the calculation.



**Parameters**

<i>x_begin, y_begin</i>	Coordinates where the line starts.
<i>x_eind, y_eind</i>	Coordinates where the line ends.
<i>colour</i>	colour of the line

**Return values**

<i>Error</i>	
--------------	--

**Author**

Osman Pekcan

**Note**

Possible to optimize this function, since there was a finite amount of time this project needed to handed in.

**Warning**

It is possible to relocate the derivative calculation but when the x offset is zero, the calculate would have a division by zero.

Here is the caller graph for this function:



## 7.2 API.h

[Go to the documentation of this file.](#)

```
1
8 #include <stdio.h>
9 #include <stdint.h>
10 #include <math.h>
11 #include <string.h>
12 #include <stdarg.h>
13 #include <stdlib.h>
14
15 //VGA
16 #include "stm32_ub_vga_screen.h"
17
18 //DEBUGS
19 //define DEBUG_FONT_NAME
20 //define DEBUG_LINE_OFFSET
21 //define DEBUG_LINE_DERIVATIVE
22 //define DEBUG_LINE_VAR
23 //define DEBUG_FIGURE_COR
24 //define DEBUG_CIRCLE_PLAATS
25 //define DEBUG_RECT_BOUNDS
```

```

26 //define DEBUG_LIJN_DIKTE
27 //define DEBUG_BITMAP_SIZE
28 //define DEBUG_ANGLE_OF_CIRCLE
29 //define DEBUG_CORNER_PARALLELOGRAM
30
31 #ifndef __weak
32     #define __weak    __attribute__((weak))
33 #endif /* __weak */
34
35 //Defines
36 #define BEGGINING_OF_BITMAP      3    //The first 3 bytes are reserved for Y and X of screen
37 #define BITMAP_WIDTH_HIGH        2    //The 3rd byte is the Width high
38 #define BITMAP_WIDTH_LOW         1    //The 2nd byte is the Width Low
39 #define BITMAP_LENGTH            0    //The first byte is the length of the bitmap.
40 #define FONT_TYPES               6    //The font type, from byte 6 is Consolas, else font type is Arial
41 #define BYTE_SIZE                8    //The size of a byte
42 #define FONT_NUMBER              2    //The amount of fonts, this can be expended.
43 #define FONT_NAME_SIZE           20   //The Size of fontname.
44 #define FONT_SIZE                2    //Size of font, 1=8 and 2=32
45 #define FONT_STYLE               2    //Style of font, 0=standard, 1=Italic and 2=Bold
46 #define SPACE_BETWEEN_LETTER    2    //The space between the letter.
47 #define LETTER_BITS              31   //Letter bits
48 #define LETTER_WIDTH_BITS        32   //Character bits
49 #define FONT_SIZE_POSITION       3    //Position of font size in the array for each fonts.
50 #define COLOURMAX                255  //colour between 0 and 255
51 #define COLOURMIN                0    //Min colour digit.
52 #define LINE_WIDTH               10   // <----*
53 #define X_BEGIN                  0
54 #define Y_BEGIN                  0
55 #define FIGURE_POINTS            5
56 #define RADIUS_INCREMENT_CIRCLE 360 //increment division of circle function
57 #define HALF_CIRCLE_DEGREE      180 // angle of halve circle in degree
58
59 __weak void drawPixel(uint16_t xp, uint16_t yp, uint8_t colour);
83 int drawBitmap(int nr, uint8_t x_lup, uint8_t y_lup);
84
113 int drawText(int x, int y, uint8_t colour, char tekst[], char fontname[], uint8_t fontgrootte, uint8_t
    fontstyle, uint8_t spaces, uint8_t linefeed);
114
143 int myLijntekenaar(uint16_t x_begin, uint16_t y_begin, uint16_t x_eind, uint16_t y_eind, uint8_t colour);
144
161 int drawLines( uint16_t x_begin, uint16_t y_begin, uint16_t x_eind, uint16_t y_eind, uint8_t
    colour, uint8_t lijn_dikte, uint8_t mid_lijn);
162
180 int drawRect(uint16_t x_pos, uint16_t y_pos, uint16_t length, uint16_t width, uint8_t colour, uint8_t
    filled, uint8_t rectborder, uint8_t border_colour);
181
200 int drawCircle(uint16_t x_pos, uint16_t y_pos, uint8_t radius, uint8_t colour, uint8_t lradius);
217 int drawFigure(uint8_t colour, uint8_t lijn_dikte, uint8_t nr_pointsgiven, ...);
218 int drawCircleplus(uint16_t x_pos, uint16_t y_pos, uint8_t l_radius, uint8_t h_radius, uint16_t
    l_angle, uint16_t h_angle, uint8_t colour);
219 int drawParallelogram(uint16_t x_pos, uint16_t y_pos, uint16_t length, uint16_t width, uint8_t angle, uint8_t
    colour);
220

```

## 7.3 Core/Inc/bitmap.h File Reference

: bitmap arrays to be printed on the screen

```
#include <stdint.h>
```

### Variables

- const uint8\_t `smiley_map` []
- const uint8\_t `angry_map` []
- const uint8\_t `left_map` []
- const uint8\_t `right_map` []
- const uint8\_t `up_map` []
- const uint8\_t `down_map` []
- const uint8\_t `smiley_map_2` []

- `const uint8_t angry_map_2 []`
- `const uint8_t right_map_old []`
- `const uint8_t smiley_map_new []`
- `const uint8_t * bitmaps [] = { smiley\_map, angry_map, left_map, right_map, up_map, down_map, smiley↵  
_map_2, angry_map_2, right_map_old, smiley_map_new }`

### 7.3.1 Detailed Description

: bitmap arrays to be printed on the screen

### Author

: Djalil

### 7.3.2 Variable Documentation

### 7.3.2.1 smiley\_map

```
const uint8_t smiley_map[]
```

Smiley\_map (50%) = 0 Angry\_map (50%) = 1 Left\_map = 2 Right\_map = 3 Up\_map = 4 Down\_map = 5 Smiley\_↵  
map 2 (fullscreen)= 6 Angry\_map 2 (fullscreen)= 7

## 7.4 bitmap.h

[Go to the documentation of this file.](#)

[illegible]

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen





Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

```

    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfb, 0xc9, 0xed, 0xe4, 0xc0,
    0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xc0, 0xc0, 0xe0,
    0xed, 0xe9, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
198    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xfb, 0xed, 0xc0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xe0, 0xc0, 0xe9, 0xfb, 0xfb, 0xf7,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
199 } ;
200
201 //-----
202 //ArrowLeft ← 45x40 pixels in HEX: 0x2D x 0x28
203 //-----
204 const uint8_t left_map[] = {
205     /*Pixel format: Red: 3 bit, Green: 3 bit, Blue: 2 bit*/
206     0x28, 0x00, 0x2D,    //Bitmap size first byte is the y boundary (VGA), 2nd and 3rd bytes are the
        x boundary (VGA).
207
208     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
209     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
210     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
211     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
212     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0x97, 0x0a, 0xdf, 0xbb, 0xff, 0x0a, 0xdf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
213     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0x53, 0x0a, 0xff, 0x77, 0x77, 0x97, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
214     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0x0a, 0xff, 0x53, 0x77, 0xdf, 0x97, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
215     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0x33, 0x97, 0xdf, 0xff, 0x97, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
216     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0x0e, 0x9b, 0xbb, 0xff, 0x9b, 0x97, 0x0e, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
217     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xdf, 0x9b, 0xff, 0xdf, 0xbb, 0x97, 0x2e, 0x77, 0x97, 0x97, 0x97, 0x97, 0x97, 0x97, 0x97, 0x97,
    0x97, 0x97, 0x97, 0x97, 0x97, 0x97, 0xbb, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
218     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xdf, 0x0a, 0x77, 0xff, 0x0e, 0xdf,
    0x9b, 0xdf, 0xff, 0x77, 0xff, 0x97, 0x33, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e,
    0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0a, 0x97, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
219     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xbb, 0x0a, 0xbb, 0xff, 0x0e, 0xff, 0xff,
    0x97, 0xdf, 0x97, 0xbb, 0x9b, 0x77, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0a, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
220     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x77, 0x0a, 0xdf, 0xbb, 0x53, 0xdf, 0xdf, 0xdf,
    0x77, 0xdf, 0x9b, 0xbb, 0x33, 0x53, 0x53, 0x32, 0x33, 0x53, 0x33, 0x53, 0x53, 0x33, 0x53,
    0x33, 0x0e, 0x57, 0x53, 0x53, 0x53, 0xbb, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff,
221     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x53, 0x0a, 0xff, 0x97, 0x53, 0xdf, 0xbb, 0xdf, 0x77,
    0x9b, 0xbb, 0xbb, 0xff, 0x9b, 0xff, 0xbb, 0xbb, 0xff, 0xdf, 0xdf, 0xbb, 0xbb, 0xbb, 0xff,
    0xbb, 0xbb, 0xbb, 0xff, 0xdf, 0xbb, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff,
222     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0e, 0x0a, 0xff, 0x53, 0x97, 0xbb, 0xbb, 0xdf, 0xbb, 0xff,
    0xbb, 0xdf, 0x9b, 0xff, 0xbb, 0xdf, 0x53, 0xdf, 0x9b, 0xbb, 0xdf, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff,
    0xdf, 0xdf, 0xbb, 0x97, 0xbb, 0xbb, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff,
223     0xff, 0xff, 0xff, 0xff, 0xff, 0x0e, 0x0e, 0xff, 0x0e, 0x2e, 0xbb, 0xdf, 0xbb, 0xdf, 0x9b, 0xdf, 0xbb,
    0x77, 0xbb, 0xbb, 0xff, 0x9b, 0x97, 0xdf, 0xbb, 0xdf, 0xdf, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff,
    0xdf, 0xdf, 0xff, 0xdf, 0xdf, 0xdf, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff,
224     0xff, 0xff, 0xff, 0xff, 0x0e, 0x0e, 0xff, 0x0e, 0xbb, 0xff, 0xdf, 0x9b, 0xff, 0xbb, 0x9b, 0xff,
    0xff, 0x9b, 0xbb, 0xff, 0x9b, 0xbb, 0x97, 0x9b, 0xbb, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff,
    0x9b, 0xff, 0xff, 0x97, 0xbb, 0xbb, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff,
225     0xff, 0xff, 0xff, 0x0e, 0x33, 0xff, 0x0e, 0xbb, 0xff, 0xbb, 0xbb, 0xdf, 0xbb, 0xdf, 0xdf,
    0xff, 0x9b, 0xdf, 0xbb, 0xff, 0x77, 0xdf, 0x9b, 0x77, 0xbb, 0xbb, 0xbb, 0xdf, 0xff, 0xdf, 0xdf,
    0xbb, 0xff, 0x9b, 0xbb, 0xbb, 0x97, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff,
226     0xff, 0xff, 0x0e, 0x2e, 0xff, 0x0e, 0xbb, 0xff, 0xbb, 0xdf, 0x77, 0xdf, 0xbb, 0xdf, 0x77, 0xdf,
    0xdf, 0xbb, 0xbf, 0x9b, 0xbb, 0x9b, 0xdf, 0xdf, 0xbb, 0x9b, 0xff, 0xbb, 0xbb, 0xbb, 0xdf, 0xbb, 0x97,
    0xff, 0xbb, 0x97, 0xbb, 0xdf, 0xbb, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff,
227     0xff, 0x77, 0x0a, 0xff, 0x0a, 0xdf, 0xbb, 0xbb, 0xff, 0xdf, 0x97, 0xdf, 0xdf, 0xbb,
    0x97, 0xff, 0x77, 0xbb, 0xdf, 0xbb, 0xbb, 0xdf, 0xff, 0x9b, 0xff, 0xbb, 0xdf, 0xff, 0xdf, 0xdf,
    0x9b, 0xbb, 0xff, 0x97, 0xdf, 0xbb, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff,
228     0xff, 0xff, 0x0a, 0x77, 0xdf, 0x2e, 0xff, 0xbb, 0xff, 0x9b, 0xff, 0x9b, 0xbb, 0xff, 0xff, 0xbb,
    0xff, 0xbb, 0x97, 0xff, 0x53, 0xbb, 0x77, 0xff, 0xbb, 0xdf, 0xdf, 0x9b, 0x9b, 0xff, 0xbb, 0xbb, 0xbb,
    0xdf, 0xbb, 0x97, 0xbb, 0xbb, 0xff, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff,
229     0xff, 0xff, 0xff, 0x0a, 0x9b, 0xff, 0x0e, 0xbb, 0xdf, 0x9b, 0xdf, 0x9b, 0xff, 0xdf, 0xbb, 0xbb,
    0xff, 0xbb, 0xdb, 0x77, 0xbb, 0x9b, 0xdf, 0xdf, 0xff, 0xdf, 0xdf, 0x53, 0xdf, 0xdf, 0xbb, 0xdf, 0xdf,
    0xdf, 0xdf, 0x9b, 0x9b, 0xdf, 0x77, 0xbb, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff,
230     0xff, 0xff, 0xff, 0xff, 0x0a, 0x77, 0xff, 0x0e, 0xbb, 0xdf, 0xbb, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e,
    0xdf, 0xbb, 0xbb, 0xbb, 0xdf, 0xbb, 0xdf, 0xdf, 0x9b, 0xbb, 0xbb, 0xdf, 0xbb, 0xbb, 0xdf, 0xdf,

```

```

0x9b, 0xdf, 0xbb, 0x77, 0x9b, 0xdf, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff,
231 0xff, 0xff, 0xff, 0xff, 0xff, 0x0a, 0x53, 0xff, 0x0e, 0xdf, 0x77, 0xff, 0xff, 0x33, 0x77, 0xdf,
0xff, 0xdf, 0x9b, 0xdf, 0xbb, 0xbb, 0x77, 0xff, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xdf, 0xdf, 0x9b, 0xdf, 0xff,
0xdf, 0xbb, 0xbb, 0xff, 0x77, 0xff, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff,
232 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0e, 0x0e, 0xff, 0x0e, 0xdf, 0xdf, 0xbb, 0x9b, 0xdf, 0xdf,
0xbb, 0x9b, 0xff, 0xbb, 0xbb, 0xbb, 0xdf, 0xdf, 0xff, 0x97, 0xdf, 0xdf, 0xdf, 0x97, 0xbb, 0xbb, 0xdf,
0xff, 0xdf, 0x77, 0xff, 0xbb, 0x97, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff,
233 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0e, 0x0e, 0xff, 0x33, 0x77, 0x97, 0x97, 0xdf, 0x9b,
0x77, 0xff, 0xdf, 0xff, 0xdf, 0xdf, 0xdf, 0xbb, 0xff, 0xdf, 0x97, 0xff, 0x9b, 0xbb, 0x9b, 0xdf, 0xff,
0xff, 0x77, 0xff, 0xdf, 0xff, 0xdf, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff,
234 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x53, 0x0a, 0xff, 0x53, 0x77, 0xff, 0xdf, 0x77,
0xbb, 0xff, 0x9b, 0xff, 0xff, 0x77, 0x53, 0x9b, 0x77, 0x53, 0x77, 0x77, 0x97, 0x77, 0x97, 0x97,
0x77, 0x33, 0x97, 0x97, 0x77, 0x77, 0x9b, 0x0e, 0xdf, 0xff, 0xff, 0xff,
235 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x53, 0x0a, 0xff, 0x53, 0x77, 0xff, 0xdf, 0x77,
0xff, 0xdf, 0xdf, 0xdf, 0x33, 0xdb, 0xbb, 0xbb, 0xbb, 0xbb, 0xbb, 0xbb, 0xbb, 0xbb, 0xbb,
0xbb, 0xbb, 0xbb, 0xbb, 0xbb, 0xbb, 0xff, 0x0a, 0xff, 0xff, 0xff,
236 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0a, 0xdf, 0xbb, 0x0e, 0xff,
0xff, 0xbb, 0xdf, 0xff, 0xbb, 0x97, 0x53, 0x53, 0x53, 0x53, 0x53, 0x53, 0x53, 0x53, 0x53,
0x53, 0x53, 0x53, 0x53, 0x53, 0x77, 0x0e, 0x73, 0xff, 0xff, 0xff, 0xff,
237 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0a, 0xbb, 0xff, 0x0e,
0xff, 0xdf, 0xdf, 0xff, 0xbb, 0x97, 0x33, 0x32, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x33,
0x33, 0x33, 0x33, 0x33, 0x33, 0x33, 0x77, 0xff, 0xff, 0xff, 0xff,
238 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0a, 0x77, 0xff,
0x0e, 0xbb, 0xff, 0xbb, 0xbb, 0x97, 0x2e, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
239 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0a, 0x53,
0xff, 0x0e, 0xdf, 0xbb, 0xbb, 0x97, 0x2e, 0xdf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
240 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0e,
0x2e, 0xff, 0x0e, 0xbb, 0x9b, 0x9b, 0x2e, 0xdf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
241 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0x0e, 0xff, 0x33, 0x53, 0x9b, 0x2e, 0xdf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
242 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0x53, 0x0a, 0xff, 0x53, 0xdf, 0x0e, 0xdf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
243 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0x53, 0x0a, 0xff, 0xbb, 0x0a, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
244 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0x73, 0x0a, 0x0a, 0x0a, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
245 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
246 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
247 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
248 } ;
249 //-----
250 //ArrowRight → 45x40 pixels in HEX: 0x2D x 0x28
251 //-----
252 const uint8_t right_map[] = {
253     /*Pixel format: Red: 3 bit, Green: 3 bit, Blue: 2 bit*/
254     0x28, 0x00, 0x2D,    //Bitmap size first byte is the y boundary (VGA), 2nd and 3rd bytes are the
        x boundary (VGA) .
255
256     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
257 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
258 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
259 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
260 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0x0a, 0xbb, 0xff, 0x0a, 0x53, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
261 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0x0e, 0xdf, 0x53, 0xff, 0x0a, 0x53, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
262 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xdf, 0x2e, 0x9b, 0x53, 0x33, 0xff, 0x0e, 0x0e, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
263 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xdf, 0x2e, 0x9b, 0x9b, 0xbb, 0x0e, 0xff, 0x2e, 0x0e, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
264 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xdf, 0x2e, 0x97, 0xbb, 0xbb, 0xdf, 0x0e, 0xff, 0x53, 0x0a, 0xff, 0xff,

```

Generated by Doxygen

```

    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
294     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
295     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
296 } ;
297
298 //-----
299 //ArrowUp ↑ 40x45 pixels in HEX: 0x28 x 0x2D
300 //-----
301 const uint8_t up_map[] = {
302     /*Pixel format: Red: 3 bit, Green: 3 bit, Blue: 2 bit*/
303     0x2D, 0x00, 0x28, //Bitmap size first byte is the y boundary (VGA), 2nd and 3rd bytes are the x
        boundary (VGA).
304
305     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
306     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0x77, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
307     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0x0a, 0x0a, 0x0e, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
308     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0x0a, 0x77, 0xff, 0x2e, 0x0e, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
309     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0x0a, 0x9b, 0xdf, 0x0a, 0xff, 0x33, 0x0e, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
310     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0x0a, 0x77, 0xff, 0x2e, 0xdf, 0x0e, 0xff, 0x0e, 0x0e, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
311     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0e,
    0x53, 0xff, 0x0e, 0xff, 0xbb, 0xbb, 0x0e, 0xff, 0x0e, 0x0e, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
312     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0e, 0x0e,
    0xff, 0x0e, 0xbb, 0xbb, 0xbb, 0xff, 0xbb, 0x0e, 0xff, 0x0a, 0x53, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
313     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x53, 0x0e, 0xff,
    0x0e, 0xbb, 0xdf, 0xff, 0xff, 0xbb, 0xff, 0xbb, 0x2e, 0xff, 0x0a, 0x77, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
314     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x53, 0x0a, 0xff, 0x0e,
    0xdf, 0xdf, 0x9b, 0x9b, 0xdf, 0xdf, 0xbb, 0xff, 0xbb, 0x53, 0xff, 0x0a, 0xbb, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
315     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0a, 0xff, 0x33, 0xdf,
    0x77, 0xbb, 0xdf, 0xff, 0x97, 0x77, 0xbb, 0xdf, 0xdf, 0x97, 0x97, 0xdf, 0x0a, 0xdf, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
316     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xbb, 0x0a, 0xff, 0x53, 0x77, 0xdf,
    0xff, 0xbb, 0x9b, 0x9b, 0xdf, 0xdf, 0xbb, 0x9b, 0xdf, 0xbb, 0x53, 0xbb, 0xbb, 0x0a, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
317     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0a, 0xdf, 0x97, 0x77, 0x97, 0xbb,
    0xff, 0xdf, 0xff, 0xbb, 0x97, 0xbb, 0xdf, 0xff, 0x9b, 0x9b, 0xdf, 0x53, 0xff, 0x77, 0x0a, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
318     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0a, 0xbb, 0xbb, 0x57, 0xff, 0x97, 0x9b,
    0x33, 0xff, 0xdf, 0xff, 0xdf, 0xdf, 0xbb, 0xbb, 0xdf, 0xdf, 0xbb, 0xdf, 0x0e, 0xff, 0x53, 0x0e, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
319     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0a, 0x77, 0xff, 0x0e, 0xbb, 0xdf, 0xdf, 0xdf,
    0x77, 0x57, 0xbb, 0xff, 0xdf, 0x77, 0xdf, 0x9b, 0xdf, 0xbb, 0xdf, 0xdf, 0xff, 0x0e, 0xff, 0x2e, 0x0e,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
320     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0e, 0x53, 0xff, 0x0e, 0xff, 0xdf, 0x77, 0x9b, 0xdf,
    0xdf, 0xdf, 0xbb, 0xbb, 0xbb, 0xdf, 0xdf, 0xff, 0xbb, 0xff, 0x77, 0xdf, 0xff, 0xdf, 0x0e, 0xff, 0x0e,
    0x53, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
321     0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0e, 0x2e, 0xff, 0x0e, 0xff, 0xff, 0xff, 0xff, 0xbb, 0x77, 0xbb,
    0xff, 0xdf, 0xff, 0xff, 0x97, 0xdf, 0xff, 0xff, 0x77, 0xbb, 0x9b, 0x77, 0x97, 0x9b, 0xdf, 0x0e, 0xff,
    0x0a, 0x53, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
322     0xff, 0xff, 0xff, 0xff, 0xff, 0x53, 0x0e, 0xff, 0x0e, 0xbb, 0xff, 0xbb, 0xbb, 0xff, 0xff, 0x9b,
    0xdf, 0xbb, 0xbb, 0xff, 0xff, 0x9b, 0x9b, 0xdf, 0xbb, 0xdf, 0xdf, 0xdf, 0x9b, 0x9b, 0x33,
    0xff, 0x0a, 0x97, 0xff, 0xff, 0xff, 0xff, 0xff,
323     0xff, 0xff, 0xff, 0xff, 0x53, 0x0a, 0xff, 0x0e, 0xdf, 0xff, 0xdf, 0xbb, 0xdf, 0x9b, 0xdf, 0xff,
    0x9b, 0xbb, 0xbb, 0x97, 0x77, 0xbf, 0xdf, 0xbb, 0xbb, 0x9b, 0xbb, 0x9b, 0x97, 0xff, 0xff, 0xbb, 0x97,
    0x53, 0xff, 0x0a, 0xbb, 0xff, 0xff, 0xff, 0xff,
324     0xff, 0xff, 0xff, 0x73, 0x0a, 0xff, 0x33, 0xbb, 0xbb, 0xbb, 0xff, 0xff, 0xdf, 0xff, 0x77, 0xbb,
    0xdf, 0xbb, 0x77, 0xff, 0xbb, 0x9b, 0xbb, 0xff, 0xff, 0xff, 0x77, 0xbb, 0xbb, 0x77, 0xdf, 0xff, 0xdf,
    0x77, 0x77, 0xdf, 0x0a, 0xbb, 0xff, 0xff,
325     0xff, 0xff, 0xff, 0x0a, 0xff, 0x53, 0x53, 0x9b, 0xbb, 0xbb, 0xbb, 0xbb, 0xff, 0xff, 0xff, 0xbb,
    0xbb, 0xdf, 0xbb, 0x53, 0xdf, 0xbb, 0xff, 0x9b, 0x9b, 0xbb, 0xff, 0xbb, 0x9b, 0x9b, 0x9b, 0x9b,
    0xdf, 0x77, 0xbb, 0x9b, 0x0a, 0xff, 0xff, 0xff,
326     0xff, 0xff, 0xff, 0x0a, 0xbb, 0xdf, 0x9b, 0x9b, 0x97, 0x97, 0x97, 0x97, 0x33, 0x77, 0xdf, 0xbb,
    0xbb, 0xbb, 0x9b, 0xbb, 0x9b, 0x77, 0xbb, 0xff, 0xdf, 0x9b, 0x33, 0x77, 0x97, 0x97, 0x97, 0x97,
    0x97, 0x97, 0xff, 0x53, 0x0e, 0xff, 0xff,
327     0xff, 0xff, 0xff, 0xbb, 0x0a, 0x0e, 0x2e, 0x2e, 0x2e, 0x2e, 0x33, 0x97, 0xdb, 0x53, 0xdf, 0xdf,
    0x77, 0xdf, 0xdf, 0x77, 0xbb, 0xdf, 0xdf, 0x9b, 0x97, 0xff, 0xff, 0x53, 0xff, 0x33, 0x2e, 0x0e, 0x0e,
    0x0e, 0x0e, 0x0a, 0x0e, 0xff, 0xff, 0xff,
328     0xff, 0xff, 0xff, 0xff, 0xff, 0xdf, 0xdf, 0xdf, 0xdf, 0xff, 0x32, 0x53, 0xbb, 0x9b, 0xbb, 0xff,

```





```

0x0a, 0x0a, 0x0e, 0x0a, 0x0e, 0x0a, 0x0e, 0x0e, 0x0a, 0x0a, 0x0e, 0x06, 0x53, 0xff, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
363 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xbb, 0x0a, 0xff, 0xbb, 0x97, 0x97, 0x97,
0x97, 0x97, 0x97, 0x97, 0x97, 0x97, 0x97, 0x9b, 0x97, 0x97, 0xff, 0x0e, 0x73, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
364 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x53, 0xbb, 0xdf, 0xdf,
0xbb, 0x97, 0xdf, 0xdf, 0xff, 0x77, 0xff, 0xff, 0x77, 0xdf, 0x77, 0x9b, 0x77, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
365 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x53, 0xdf, 0xbb, 0xdf,
0xbf, 0xbb, 0xdf, 0xdf, 0xbb, 0xdf, 0x9b, 0x77, 0xbb, 0xff, 0x77, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
366 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x52, 0xff, 0x77, 0xdf,
0x77, 0xbb, 0xbb, 0x77, 0xdf, 0x9b, 0x77, 0xff, 0xff, 0xdf, 0x77, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
367 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x73, 0x9b, 0xbb, 0xff,
0xff, 0x97, 0x77, 0xff, 0x77, 0x97, 0xbb, 0xbb, 0x53, 0xff, 0x97, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
368 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x0e, 0xbb, 0xdf, 0xbb,
0xff, 0xff, 0xdf, 0xbb, 0xbb, 0xdf, 0xff, 0xbb, 0xdf, 0x53, 0x33, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
369 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x2e, 0xbb, 0xdf, 0xdf,
0x9b, 0xbb, 0xff, 0x97, 0xff, 0xdf, 0x97, 0xdb, 0xff, 0xff, 0x57, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
370 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x53, 0xff, 0x9b, 0x77,
0xff, 0xdf, 0x77, 0xdf, 0xbb, 0xdf, 0xdf, 0xff, 0xdf, 0xff, 0x97, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
371 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x2e, 0x9b, 0xff, 0xff,
0x9b, 0xdf, 0xbb, 0xdf, 0xbb, 0xdf, 0xdf, 0xdf, 0x9b, 0xdf, 0x97, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
372 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x53, 0xbb, 0xbb, 0xbb,
0xbb, 0xff, 0xff, 0xff, 0xbb, 0xbb, 0xbb, 0x97, 0xbb, 0x9b, 0x57, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
373 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x53, 0xbb, 0xbb, 0xdf,
0xff, 0xdf, 0xbb, 0xbb, 0xff, 0xdf, 0xbb, 0xdf, 0x97, 0xbb, 0x77, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
374 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x53, 0xdf, 0xbb, 0x9b,
0x77, 0xbb, 0xbb, 0xbb, 0x97, 0xdf, 0xbb, 0xdf, 0xdf, 0x77, 0x97, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
375 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x2e, 0xdf, 0xdf, 0xff,
0xbb, 0xbb, 0xbb, 0xff, 0x9b, 0x33, 0xbb, 0xdf, 0xdf, 0xff, 0x77, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
376 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x73, 0xdf, 0x9b, 0xdf,
0x9b, 0xbb, 0xff, 0x77, 0xdf, 0xdf, 0xbb, 0x77, 0xdf, 0x77, 0x97, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
377 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x52, 0xff, 0x9b, 0xdf,
0x9b, 0xdf, 0x9b, 0xff, 0xbb, 0xdf, 0x97, 0xff, 0x77, 0xdf, 0x33, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
378 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x97, 0x0e, 0xff, 0x2e, 0xdf, 0xbb, 0xbb,
0xbb, 0x77, 0xbb, 0xdf, 0xbb, 0xff, 0xdf, 0x9b, 0xff, 0xff, 0x77, 0xbb, 0x53, 0x33, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
379 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xdf, 0xdf, 0xdf, 0xdf, 0xff, 0x77, 0x0e, 0xff, 0x53, 0xbb, 0x2f, 0xff,
0x9b, 0x97, 0xdf, 0x9b, 0xff, 0xdf, 0xdf, 0xff, 0xdf, 0xbb, 0x9b, 0xbb, 0x53, 0x2e, 0xff, 0xdf, 0xdf,
0xdf, 0xdf, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
380 0xff, 0xff, 0xff, 0x0a, 0x0a, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0xff, 0x53, 0xff, 0xff, 0x77,
0x77, 0xdf, 0xdf, 0xbf, 0x57, 0xdf, 0xdf, 0x53, 0xff, 0xdf, 0x33, 0xbb, 0x97, 0x32, 0x0e, 0x0e, 0x0e,
0x2e, 0x0a, 0x06, 0xdb, 0xff, 0xff, 0xff,
381 0xff, 0xff, 0x0a, 0x53, 0xff, 0x97, 0x97, 0x97, 0x97, 0x97, 0x97, 0x73, 0x32, 0x97, 0xdf, 0xff,
0xdf, 0x77, 0x97, 0xbb, 0xbb, 0x97, 0xbb, 0xdf, 0xbb, 0xdf, 0x77, 0x2e, 0x97, 0x97, 0x97, 0x97, 0x97,
0x97, 0xdb, 0xdf, 0x06, 0xff, 0xff, 0xff,
382 0xff, 0xff, 0x0a, 0xbb, 0xbb, 0x97, 0xff, 0xff, 0x9b, 0xbb, 0xff, 0x9b, 0xbb, 0xff, 0xbb, 0xdf, 0xbb, 0xdf, 0xbb,
0x9b, 0xff, 0xbb, 0xff, 0x33, 0xbb, 0xdf, 0xbb, 0xbb, 0xff, 0xff, 0xff, 0xbb, 0xdf, 0xbb, 0xdf, 0xbb,
0x77, 0x33, 0xff, 0x0a, 0xff, 0xff, 0xff,
383 0xff, 0xff, 0xdf, 0xdf, 0x06, 0xff, 0x73, 0x77, 0xdf, 0xff, 0xdf, 0x57, 0xbb, 0xdf, 0x57, 0xff, 0xff,
0xff, 0xbb, 0x9b, 0xbb, 0xff, 0x77, 0xdf, 0xdf, 0xbb, 0x77, 0xff, 0xdf, 0xff, 0xff, 0xbb, 0xbb, 0xbb,
0x0e, 0xff, 0x0a, 0x53, 0xff, 0xff, 0xff,
384 0xff, 0xff, 0xff, 0xdf, 0x06, 0xff, 0x32, 0x97, 0xbb, 0xff, 0xff, 0x77, 0x97, 0xbb, 0x9b, 0xbb,
0xbb, 0xdf, 0xbb, 0x77, 0x77, 0xbb, 0xbb, 0x97, 0xff, 0xff, 0x97, 0xdf, 0xbb, 0xdf, 0xff, 0xff, 0x0a,
0xff, 0x0a, 0x53, 0xff, 0xff, 0xff, 0xff,
385 0xff, 0xff, 0xff, 0x97, 0x0a, 0xff, 0x0e, 0xbb, 0x9b, 0xdf, 0xdf, 0xdf, 0xbb, 0xdf, 0xbb,
0x97, 0x9b, 0x9b, 0xff, 0x9b, 0xbb, 0x9b, 0xff, 0x9b, 0xff, 0x9b, 0x9b, 0xff, 0xbb, 0x0a, 0xff,
0x0a, 0x53, 0xff, 0xff, 0xff, 0xff, 0xff,
386 0xff, 0xff, 0xff, 0xff, 0xff, 0x33, 0x0a, 0xff, 0x0a, 0xdf, 0x9b, 0x77, 0x77, 0x9b, 0xbb, 0x77,
0xff, 0xff, 0xdf, 0x77, 0xff, 0xbf, 0xff, 0xbb, 0x77, 0xbb, 0xff, 0xff, 0xff, 0x0a, 0xff, 0x2e,
0x0e, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
387 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x2e, 0x0e, 0xff, 0x0a, 0xdf, 0xff, 0xdf, 0x77, 0xff, 0x9b,
0xff, 0xbb, 0xff, 0x9b, 0xbb, 0xbb, 0xdf, 0xdf, 0xdf, 0x9b, 0x77, 0xdf, 0xff, 0x0a, 0xff, 0x33, 0x0e,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
388 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0e, 0x2e, 0xff, 0x0a, 0xff, 0xdf, 0xff, 0x9b, 0xdf,
0x77, 0xdf, 0x53, 0xff, 0xff, 0xdf, 0x53, 0x97, 0xdf, 0xff, 0xff, 0xbb, 0x0e, 0xff, 0x57, 0x0a, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
389 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0e, 0x33, 0xff, 0x0e, 0xdf, 0xbb, 0xdf, 0xdf,
0xbb, 0xbb, 0xff, 0xdf, 0xff, 0xbb, 0xff, 0x0e, 0x9b, 0x77, 0xff, 0x73, 0xdb, 0x9b, 0x0a, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
390 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0a, 0x57, 0xff, 0x2e, 0xff, 0x9b, 0x97,
0xff, 0xdf, 0xbb, 0x77, 0xbb, 0xff, 0xdf, 0xff, 0xdf, 0x77, 0x97, 0x77, 0xff, 0x06, 0xff, 0xff, 0xff,
0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
391 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x0a, 0x9b, 0xdb, 0x53, 0xbb, 0xdf,

```

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen



Generated by Doxygen

Generated by Doxygen

Generated by Doxygen





Generated by Doxygen

Generated by Doxygen

Generated by Doxygen







Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen



Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen



Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen



Generated by Doxygen

Generated by Doxygen

[illegible]

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen



Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

[illegible]





Generated by Doxygen



Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

[illegible]





Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

[illegible]

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen



Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

[illegible]

Generated by Doxygen





Generated by Doxygen

[illegible]

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen



Generated by Doxygen



Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen



Generated by Doxygen





Generated by Doxygen

Generated by Doxygen

[illegible]

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen









Generated by Doxygen

Generated by Doxygen



Generated by Doxygen





Generated by Doxygen





Generated by Doxygen



Generated by Doxygen

Generated by Doxygen

Generated by Doxygen



Generated by Doxygen





Generated by Doxygen





Generated by Doxygen

Generated by Doxygen

Generated by Doxygen

Generated by Doxygen













[illegible]









Generated by Doxygen



Generated by Doxygen



Generated by Doxygen



Generated by Doxygen







Generated by Doxygen

Generated by Doxygen



Generated by Doxygen



Generated by Doxygen





Generated by Doxygen



Generated by Doxygen



Generated by Doxygen

Generated by Doxygen

Generated by Doxygen



Generated by Doxygen



- `#define ERROR_LINE_WIDTH_OUT_OF_RANGE 12`
- `#define ERROR_RECT_OUT_OF_RANGE 13`
- `#define ERROR_RECT_COLOUR_OUT_OF_RANGE 14`
- `#define ERROR_FILLED_OUT_OF_RANGE 15`
- `#define ERROR_CIRCLE_COLOUR_OUT_OF_RANGE 16`
- `#define ERROR_CIRCLE_OUT_OF_RANGE 17`
- `#define ERROR_FIGURE_COLOUR_OUT_OF_RANGE 18`
- `#define ERROR_FIGURE_LENGTH_OUT_OF_RANGE 19`
- `#define ERROR_COLOUR_NOT_FOUND 20`
- `#define ERROR_CIRCLE_WRONG_RADIUS 21`
- `#define ERROR_CIRCLE_OUT_OF_ANGLE 22`
- `#define ERROR_PARALLELOGRAM_OUT_OF_RANGE 23`
- `#define ERROR_PARALLELOGRAM_OUT_OF_ANGLE 24`
- `#define ERROR_FONTSTYLE_NOT_FOUND 255`
- `#define ERROR_COLOR_NOT_FOUND 254`

## Functions

- `void softonErrorHandler (uint8_t error)`

### 7.5.1 Detailed Description

: Defines for all error codes

Author

: Djalil

## 7.6 error.h

[Go to the documentation of this file.](#)

```

1
2 #include <stdint.h>
3 #include <stdio.h>
4
5 #define ERROR_BITMAP_NOT_FOUND 1
6 #define ERROR_BITMAP_OUT_OF_RANGE 2
7 #define ERROR_BITMAP_POSITION_OUT_OF_RANGE 3
8 #define ERROR_FONT_COLOUR_OUT_OF_RANGE 4
9 #define ERROR_FONT_NOT_FOUND 5
10 #define ERROR_FONT_OUT_OF_RANGE 6
11 #define ERROR_FONT_SIZE_OUT_OF_RANGE 7
12 #define ERROR_FONTSTYLE_OUT_OF_RANGE 8
13 #define ERROR_LINE_OUT_OF_RANGE 9
14 #define ERROR_LINE_LENGTH_OUT_OF_RANGE 10
15 #define ERROR_LINE_COLOUR_OUT_OF_RANGE 11
16 #define ERROR_LINE_WIDTH_OUT_OF_RANGE 12
17 #define ERROR_RECT_OUT_OF_RANGE 13
18 #define ERROR_RECT_COLOUR_OUT_OF_RANGE 14
19 #define ERROR_FILLED_OUT_OF_RANGE 15
20 #define ERROR_CIRCLE_COLOUR_OUT_OF_RANGE 16
21 #define ERROR_CIRCLE_OUT_OF_RANGE 17
22 #define ERROR_FIGURE_COLOUR_OUT_OF_RANGE 18
23 #define ERROR_FIGURE_LENGTH_OUT_OF_RANGE 19
24 #define ERROR_COLOUR_NOT_FOUND 20
25 #define ERROR_CIRCLE_WRONG_RADIUS 21
26 #define ERROR_CIRCLE_OUT_OF_ANGLE 22
27 #define ERROR_PARALLELOGRAM_OUT_OF_RANGE 23
28 #define ERROR_PARALLELOGRAM_OUT_OF_ANGLE 24
29
30 #define ERROR_FONTSTYLE_NOT_FOUND 255
31 #define ERROR_COLOR_NOT_FOUND 254
32
33 //prototypes
34 void softonErrorHandler(uint8_t error);

```

## 7.7 Core/Inc/front\_layer.h File Reference

```
#include <stdint.h>
#include "main.h"
#include "error.h"
```

### Data Structures

- struct [input\\_vars](#)

### Macros

- #define **BYTE\_BUFLen** 1
- #define **LINE\_BUFLen** 1024
- #define **CARRIAGE\_RETURN** 13 /\* carriage return char \r \*/
- #define **LINE\_FEED** 10 /\* linefeed char \n \*/

### Functions

- void [FL\\_UART\\_Init](#) ()  
*Function for initialisatie of the UART.*
- void [FL\\_Parser](#) ()
- void [error\\_on\\_screen](#) (uint8\_t error)

### Variables

- [input\\_vars](#) input

#### 7.7.1 Detailed Description

Version

1.0

#### 7.7.2 Function Documentation

### 7.7.2.1 FL\_UART\_Init()

```
void FL_UART_Init ( )
```

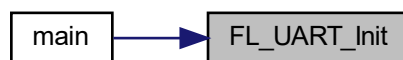
Function for initialisatie of the UART.

First the function empties out the buffer and initialist the HAL UART

Author

Tjerk ten Dam

Here is the caller graph for this function:



## 7.8 front\_layer.h

[Go to the documentation of this file.](#)

```

1
6 #include <stdint.h>
7 #include "main.h"
8 #include "error.h"
9
10
11 #define BYTE_BUFLen      1
12 #define LINE_BUFLen      1024
13 #define CARRIAGE_RETURN  13 /* carriage return char \r */
14 #define LINE_FEED        10 /* linefeed char \n */
15
16 typedef struct
17 {
18     uint8_t byte_buffer_rx[BYTE_BUFLen]; // Store the rx byte from the USART2
19     char line_rx_buffer[LINE_BUFLen];    // Buffer to hold all the bytes from rx USART2
20     int msglen;
21     volatile int char_counter;           // Counter for line_rx_buffer
22     char command_execute_flag;           /* Set = whole transmission is received, ready for processing
23                                         \
24                                         Reset = still receiving*/
24 }input_vars;
25
26 extern input_vars input;
27
28 void FL_UART_Init();
29 void FL_Parser();
30 void error_on_screen(uint8_t error);
  
```

## 7.9 Core/Inc/logic\_layer.h File Reference

: logic layer

```

#include <stdio.h>
#include <stdint.h>
#include <string.h>
#include <stdlib.h>
#include "API.h"
  
```

## Data Structures

- struct [wait\\_vars](#)

## Macros

- #define **NUMBER\_OF\_COMMANDS** 11
- #define **POINT\_OF\_FIGURE** 5
- #define **MAX\_NUMBER\_OF\_SCRIPT\_CHARACTER** 150
- #define **MAX\_SCRIPT\_COMMANDOS** 15
- #define **ASCII\_OF\_COMMA** 44
- #define **ASCII\_OF\_BACKSLASH** 92
- #define **COLOUR\_CHAR\_MAX** 50
- #define **COLOUR\_LIST** 17
- #define **NUMBER\_OF\_FONT\_STYLES** 3
- #define **NUMBER\_OF\_COLORS** 10
- #define **BUFFER\_SIZE** 200

## Functions

- int [logic\\_layer](#) (char commando[])  
*This function looks at input array and calls function the input array calls for.*
- void [HAL\\_TIM\\_PeriodElapsedCallback](#) (TIM\_HandleTypeDef \*htim)  
*Tim callback when period is elapsed.*
- void [buffer\\_init](#) ()  
*This function is for initilising buffer of scripts.*
- void [writeBuffer](#) (char buf[])  
*This function writes a array to the circular script buffer.*
- int [indexBuffer](#) (int index)  
*This function translates an index for the script buffer.*

## Variables

- [wait\\_vars](#) wait

### 7.9.1 Detailed Description

: logic layer

### 7.9.2 Function Documentation

### 7.9.2.1 buffer\_init()

```
void buffer_init ( )
```

This function is for initilising buffer of scripts.

#### Authors

Tjerk ten Dam

### 7.9.2.2 HAL\_TIM\_PeriodElapsedCallback()

```
void HAL_TIM_PeriodElapsedCallback (
    TIM_HandleTypeDef * htim )
```

Tim callback when period is elapsed.

This callback functions gets called when the timer period is elapsed for the wait function and when elapsed the wait flag gets reset.

#### Parameters

<i>*htim</i>	handle of the timer
--------------	---------------------

#### Authors

Tjerk ten Dam

### 7.9.2.3 indexBuffer()

```
int indexBuffer (
    int index )
```

This function translates an index for the script buffer.

#### Parameters

<i>index</i>	index for the script buffer
--------------	-----------------------------

#### Return values

<i>correct</i>	index for the circular buffer
----------------	-------------------------------

#### Authors

Tjerk ten Dam

#### 7.9.2.4 logic\_layer()

```
int logic_layer (
    char commando[ ] )
```

This function looks at input array and calls function the input array calls for.

First script commando's are split up and put into array: `commando_filled`. Then the function looks which script is called. If the function found the script that is called, the corresponding function is called.

#### Parameters

<i>commando</i>	This is the script commando.
-----------------	------------------------------

#### Authors

Osman Pekcan,Tjerk ten Dam

#### 7.9.2.5 writeBuffer()

```
void writeBuffer (
    char buf[ ] )
```

This function writes a array to the circular script buffer.

#### Parameters

<i>buf</i>	array to be write to the script buffer
------------	--

#### Authors

Tjerk ten Dam

## 7.10 logic\_layer.h

[Go to the documentation of this file.](#)

```
1
8 #include <stdio.h>
9 #include <stdint.h>
10 #include <string.h>
11 #include <stdlib.h>
12 #include "API.h"
```



```

13
14 #define NUMBER_OF_COMMANDS 11
15 #define POINT_OF_FIGURE 5
16 #define MAX_NUMBER_OF_SCRIPT_CHARACTER 150
17 #define MAX_SCRIPT_COMMANDS 15
18 #define ASCII_OF_COMMA 44
19 #define ASCII_OF_BACKSLASH 92
20 #define COLOUR_CHAR_MAX 50
21 #define COLOUR_LIST 17
22
23
24 #define NUMBER_OF_FONT_STYLES 3
25 #define NUMBER_OF_COLORS 10
26 // #define DEBUG_COMMANDO
27 // #define DEBUG_NUMBEROFCOMMANDO
28
29 #define BUFFER_SIZE 200
30
31
32 typedef struct
33 {
34     char buffer[BUFFER_SIZE][MAX_NUMBER_OF_SCRIPT_CHARACTER];
35     int counter;
36     int waitFlag;
37     int head;
38     int executed;
39 }wait_vars;
40
41 extern wait_vars wait;
42
43 //Prototypes
44 int logic_layer(char commando[]);
45 void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim);
46 void buffer_init();
47 void writeBuffer(char buf[]);
48 int indexBuffer(int index);
49

```

## 7.11 Core/Inc/main.h File Reference

: Header for [main.c](#) file. This file contains the common defines of the application.

```

#include "stm32f4xx_hal.h"
#include "stdint.h"
#include "stdio.h"
#include "string.h"
#include <stdlib.h>
#include "stm32_ub_vga_screen.h"

```

### Macros

- #define **VGA\_BLUE0\_Pin** GPIO\_PIN\_8
- #define **VGA\_BLUE0\_GPIO\_Port** GPIOE
- #define **VGA\_BLUE1\_Pin** GPIO\_PIN\_9
- #define **VGA\_BLUE1\_GPIO\_Port** GPIOE
- #define **VGA\_GREEN0\_Pin** GPIO\_PIN\_10
- #define **VGA\_GREEN0\_GPIO\_Port** GPIOE
- #define **VGA\_GREEN1\_Pin** GPIO\_PIN\_11
- #define **VGA\_GREEN1\_GPIO\_Port** GPIOE
- #define **VGA\_GREEN2\_Pin** GPIO\_PIN\_12
- #define **VGA\_GREEN2\_GPIO\_Port** GPIOE
- #define **VGA\_RED0\_Pin** GPIO\_PIN\_13
- #define **VGA\_RED0\_GPIO\_Port** GPIOE
- #define **VGA\_RED1\_Pin** GPIO\_PIN\_14

- `#define VGA_RED1_GPIO_Port GPIOE`
- `#define VGA_RED2_Pin GPIO_PIN_15`
- `#define VGA_RED2_GPIO_Port GPIOE`
- `#define VGA_HSYNC_Pin GPIO_PIN_11`
- `#define VGA_HSYNC_GPIO_Port GPIOB`
- `#define VGA_VSYNC_Pin GPIO_PIN_12`
- `#define VGA_VSYNC_GPIO_Port GPIOB`
- `#define FALSE 0x00`
- `#define TRUE 0xFF`

## Functions

- void `Error_Handler` (void)

*This function is executed in case of error occurrence.*

## Variables

- volatile char `container` [1024]
- volatile int `temp`
- volatile int `key`

### 7.11.1 Detailed Description

: Header for `main.c` file. This file contains the common defines of the application.

#### Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: [opensource.org/licenses/BSD-3-Clause](https://opensource.org/licenses/BSD-3-Clause)

### 7.11.2 Function Documentation

#### 7.11.2.1 Error\_Handler()

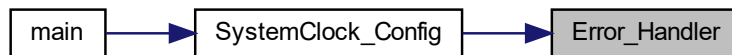
```
void Error_Handler (  
    void )
```

This function is executed in case of error occurrence.

## Return values

None	
------	--

Here is the caller graph for this function:



## 7.12 main.h

[Go to the documentation of this file.](#)

```

1 /* USER CODE BEGIN Header */
20 /* USER CODE END Header */
21
22 /* Define to prevent recursive inclusion -----*/
23 #ifndef __MAIN_H
24 #define __MAIN_H
25
26 #ifdef __cplusplus
27 extern "C" {
28 #endif
29
30 /* Includes -----*/
31 #include "stm32f4xx_hal.h"
32
33 /* Private includes -----*/
34 /* USER CODE BEGIN Includes */
35 #include "stdint.h"
36
37 #include "stdio.h"
38 #include "string.h"
39 #include <stdlib.h>
40 #include "stm32_ub_vga_screen.h"
41
42 /* USER CODE END Includes */
43
44 /* Exported types -----*/
45 /* USER CODE BEGIN ET */
46
47 /* USER CODE END ET */
48
49 /* Exported constants -----*/
50 /* USER CODE BEGIN EC */
51
52 /* USER CODE END EC */
53
54 /* Exported macro -----*/
55 /* USER CODE BEGIN EM */
56
57 /* USER CODE END EM */
58
59 /* Exported functions prototypes -----*/
60 void Error_Handler(void);
61
62 /* USER CODE BEGIN EFP */
63
64 /* USER CODE END EFP */
65
66 /* Private defines -----*/
67 #define VGA_BLUE0_Pin GPIO_PIN_8
68 #define VGA_BLUE0_GPIO_Port GPIOE
69 #define VGA_BLUE1_Pin GPIO_PIN_9
70 #define VGA_BLUE1_GPIO_Port GPIOE
71 #define VGA_GREEN0_Pin GPIO_PIN_10
72 #define VGA_GREEN0_GPIO_Port GPIOE

```

```

73 #define VGA_GREEN1_Pin GPIO_PIN_11
74 #define VGA_GREEN1_GPIO_Port GPIOE
75 #define VGA_GREEN2_Pin GPIO_PIN_12
76 #define VGA_GREEN2_GPIO_Port GPIOE
77 #define VGA_RED0_Pin GPIO_PIN_13
78 #define VGA_RED0_GPIO_Port GPIOE
79 #define VGA_RED1_Pin GPIO_PIN_14
80 #define VGA_RED1_GPIO_Port GPIOE
81 #define VGA_RED2_Pin GPIO_PIN_15
82 #define VGA_RED2_GPIO_Port GPIOE
83 #define VGA_HSYNC_Pin GPIO_PIN_11
84 #define VGA_HSYNC_GPIO_Port GPIOB
85 #define VGA_VSYNC_Pin GPIO_PIN_12
86 #define VGA_VSYNC_GPIO_Port GPIOB
87 /* USER CODE BEGIN Private defines */
88
89 #define FALSE    0x00
90 #define TRUE     0xFF
91
92 /* Struct's -----*/
93
94
95 /* Globals -----*/
96 extern volatile char container[1024];
97 extern volatile int temp;
98 extern volatile int key;
99
100 /* USER CODE END Private defines */
101
102 #ifdef __cplusplus
103 }
104 #endif
105
106 #endif /* __MAIN_H */
107
108 /***** (C) COPYRIGHT STMicroelectronics *****/

```

## 7.13 Core/Inc/stm32f4xx\_it.h File Reference

This file contains the headers of the interrupt handlers.

### Functions

- void **NMI\_Handler** (void)  
*This function handles Non maskable interrupt.*
- void **HardFault\_Handler** (void)  
*This function handles Hard fault interrupt.*
- void **MemManage\_Handler** (void)  
*This function handles Memory management fault.*
- void **BusFault\_Handler** (void)  
*This function handles Pre-fetch fault, memory access fault.*
- void **UsageFault\_Handler** (void)  
*This function handles Undefined instruction or illegal state.*
- void **SVC\_Handler** (void)  
*This function handles System service call via SWI instruction.*
- void **DebugMon\_Handler** (void)  
*This function handles Debug monitor.*
- void **PendSV\_Handler** (void)  
*This function handles Pendable request for system service.*
- void **SysTick\_Handler** (void)  
*This function handles System tick timer.*
- void **TIM2\_IRQHandler** (void)  
*This function handles TIM2 global interrupt.*

- void **USART2\_IRQHandler** (void)  
*This function handles USART2 global interrupt.*
- void **TIM5\_IRQHandler** (void)  
*This function handles TIM5 global interrupt.*
- void **DMA2\_Stream5\_IRQHandler** (void)  
*This function handles DMA2 stream5 global interrupt.*

### 7.13.1 Detailed Description

This file contains the headers of the interrupt handlers.

#### Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: [opensource.org/licenses/BSD-3-Clause](https://opensource.org/licenses/BSD-3-Clause)

## 7.14 stm32f4xx\_it.h

[Go to the documentation of this file.](#)

```
1 /* USER CODE BEGIN Header */
19 /* USER CODE END Header */
20
21 /* Define to prevent recursive inclusion -----*/
22 #ifndef __STM32F4xx_IT_H
23 #define __STM32F4xx_IT_H
24
25 #ifdef __cplusplus
26     extern "C" {
27 #endif
28
29 /* Private includes -----*/
30 /* USER CODE BEGIN Includes */
31
32 /* USER CODE END Includes */
33
34 /* Exported types -----*/
35 /* USER CODE BEGIN ET */
36
37 /* USER CODE END ET */
38
39 /* Exported constants -----*/
40 /* USER CODE BEGIN EC */
41
42 /* USER CODE END EC */
43
44 /* Exported macro -----*/
45 /* USER CODE BEGIN EM */
46
47 /* USER CODE END EM */
48
49 /* Exported functions prototypes -----*/
50 void NMI_Handler(void);
51 void HardFault_Handler(void);
52 void MemManage_Handler(void);
53 void BusFault_Handler(void);
```

```

54 void UsageFault_Handler(void);
55 void SVC_Handler(void);
56 void DebugMon_Handler(void);
57 void PendSV_Handler(void);
58 void SysTick_Handler(void);
59 void TIM2_IRQHandler(void);
60 void USART2_IRQHandler(void);
61 void TIM5_IRQHandler(void);
62 void DMA2_Stream5_IRQHandler(void);
63 /* USER CODE BEGIN EFP */
64
65 /* USER CODE END EFP */
66
67 #ifdef __cplusplus
68 }
69 #endif
70
71 #endif /* __STM32F4xx_IT_H */
72
73 /***** (C) COPYRIGHT STMicroelectronics *****/

```

## 7.15 Core/Src/API.c File Reference

: Library for Graphics.

```

#include "API.h"
#include "front_layer.h"
#include "error.h"
#include "bitmap.h"
#include "fonts/Arial_32.h"
#include "fonts/Arial_8.h"
#include "fonts/Arial_Italic_32.h"
#include "fonts/Arial_Italic_8.h"
#include "fonts/Arial_Bold_32.h"
#include "fonts/Arial_Bold_8.h"
#include "fonts/Consolas_32.h"
#include "fonts/Consolas_8.h"
#include "fonts/Consolas_Italic_32.h"
#include "fonts/Consolas_Italic_8.h"
#include "fonts/Consolas_Bold_32.h"
#include "fonts/Consolas_Bold_8.h"

```

## Functions

- `__weak void drawPixel (uint16_t xp, uint16_t yp, uint8_t colour)`  
*Weak declared Draw pixel function for if the user wants to implement his own display interface.*
- `int myLijntekenaar (uint16_t x_begin, uint16_t y_begin, uint16_t x_eind, uint16_t y_eind, uint8_t colour)`  
*this function draws a single line from a to b*
- `int drawLines (uint16_t x_begin, uint16_t y_begin, uint16_t x_eind, uint16_t y_eind, uint8_t colour, uint8_t lijn_dikte, uint8_t mid_lijn)`  
*This function draws multiple lines using mylijntekenaar function.*
- `int drawRect (uint16_t x_pos, uint16_t y_pos, uint16_t length, uint16_t width, uint8_t colour, uint8_t filled, uint8_t rectborder, uint8_t border_colour)`  
*This function draws a rectangle either filled or not.*
- `int drawCircle (uint16_t x_pos, uint16_t y_pos, uint8_t radius, uint8_t colour, uint8_t lradius)`  
*This function draws a circle.*
- `int drawFigure (uint8_t colour, uint8_t lijn_dikte, uint8_t nr_pointsgiven,...)`  
*This functions draw a figure with X amount of points.*

- int [drawBitmap](#) (int nr, uint8\_t x\_1up, uint8\_t y\_1up)  
*Draw bitmap function, this function draws the bitmap; smiley, angry and the four arrows on the screen.*
- int [drawText](#) (int x, int y, uint8\_t colour, char tekst[], char fontname[], uint8\_t fontsize, uint8\_t fontstyle, uint8\_t spaces, uint8\_t linefeed)  
*Draw text function, this function prints the text on the screen.*
- int [drawCircleplus](#) (uint16\_t x\_pos, uint16\_t y\_pos, uint8\_t l\_radius, uint8\_t h\_radius, uint16\_t l\_angle, uint16\_t h\_angle, uint8\_t colour)  
*This function draws a circle.*
- int [drawParallelogram](#) (uint16\_t x\_pos, uint16\_t y\_pos, uint16\_t length, uint16\_t width, uint8\_t angle, uint8\_t colour)  
*This function draws an parallelogram using length,width and an angle.*

## 7.15.1 Detailed Description

: Library for Graphics.

## 7.15.2 Function Documentation

### 7.15.2.1 drawBitmap()

```
int drawBitmap (  
    int nr,  
    uint8_t x_1up,  
    uint8_t y_1up )
```

Draw bitmap function, this function draws the bitmap; smiley, angry and the four arrows on the screen.

this function shows a graphics image file (bitmaps) on VGA display.

Bitmap has 6 icons; smiley, angry and the four arrows. smiley\_map=0, angry\_map=1, Arrow left=2, Arrow up= The user can choose between smiley=0, angry=1, left=2, up=3, right=4, right=5. Add more bitmapguide: Step 1: Add bitmap array into [bitmap.h](#) Step 2: At the bottom of [bitmap.h](#) change the array and add the name of bitmap.

#### Parameters

<i>nr</i>	is the bitmap number to draw.
<i>x_1up</i>	and y_1up is the position of the bitmap

#### Return values

<i>Error</i>	
--------------	--

#### Author

Djalil & Tjerk

### 7.15.2.2 drawCircle()

```
int drawCircle (
    uint16_t x_pos,
    uint16_t y_pos,
    uint8_t radius,
    uint8_t colour,
    uint8_t lradius )
```

This function draws a circle.

By using **elementary** mathematics the function draw a circle. The for loop draws points of the circle until it drew the whole circle. The for loop goes through every angle between 0 - 2 PI radius

#### Parameters

<i>x_pos,y_pos</i>	These are the x and y coordinates of the middle of the circle.
<i>radius</i>	This is the radius of the circle.
<i>colour</i>	This is the colour of the circle.
<i>lradius</i>	This is the lower boundary of radius

mathematic proof

#### Return values

Error	
-------	--

#### Author

Osman Pekcan

### 7.15.2.3 drawCircleplus()

```
int drawCircleplus (
    uint16_t x_pos,
    uint16_t y_pos,
    uint8_t l_radius,
    uint8_t h_radius,
    uint16_t l_angle,
    uint16_t h_angle,
    uint8_t colour )
```

This function draws a circle.

By using **elementary** mathematics the function draw a circle. The for loop draws points of the circle until it drew the whole circle. The for loop goes through every angle between 0 - 2 PI radius

#### Parameters

<i>x_pos,y_pos</i>	These are the x and y coordinates of the middle of the circle.
<i>radius</i>	This is the radius of the circle.
<i>angle</i>	the angle where to start and stop the circle
<i>colour</i>	This is the colour of the circle mathematic proof



## Return values

<i>Error</i>	
--------------	--

## Author

Osman Pekcan

### 7.15.2.4 drawFigure()

```
int drawFigure (
    uint8_t colour,
    uint8_t lijn_dikte,
    uint8_t nr_pointsgiven,
    ... )
```

This functions draw a figure with X amount of points.

With the function mylijntekenaar, this function draws multiple line that are chained to each other. With *va\_list* the argument after *nr\_pointsgiven* are saved in a list. Using *va\_arg* the argument are put into two array *figure\_ram\_x* and *figure\_ram\_y*. with arithmetic code the argument that represent x coordinate are put into *figure\_ram\_x*. likewise for the y coordinates. with these to arrays the function can draw its figure with a for loop.

## Parameters

<i>colour</i>	This parameter is for colour of the lines.
<i>lijn_dikte</i>	The variable that dictate how big the line width is.
<i>nr_pointgiven</i>	Locked the number of points

## Return values

<i>Error</i>	
--------------	--

## Author

Osman Pekcan

### 7.15.2.5 drawLines()

```
int drawLines (
    uint16_t x_begin,
    uint16_t y_begin,
    uint16_t x_eind,
    uint16_t y_eind,
    uint8_t colour,
```

```
uint8_t lijn_dikte,
uint8_t mid_lijn )
```

This function draws multiple lines using mylijntekenaar function.

By repeating the function mylijntekenaar, this draws multiple lines. To do this it checks the parameter which concludes the width of the sum of lines. If it is one, the function just calls the function mylijntekenaar once. Else draws the lines around the middle line(original x and y positions). For the function to know whether to draw extra line in the x axis or y axis it looks at the derivative. If the derivative is higher than one its draw from the x axis, else its draws from the y axis

#### Parameters

<i>lijn_dikte</i>	The variable that dictate how big the line width is.
<i>mid_lijn</i>	If mid_lijn is equal to one,the function draws a black line with the original coordinates.

#### See also

[myLijntekenaar\(\)](#)

#### Return values

<i>Error</i>	
--------------	--

#### Author

Osman Pekcan

Here is the call graph for this function:



#### 7.15.2.6 drawParallelogram()

```
int drawParallelogram (
    uint16_t x_pos,
    uint16_t y_pos,
    uint16_t length,
    uint16_t width,
    uint8_t angle,
    uint8_t colour )
```

This function draws an parallelogram using length,width and an angle.

## Parameters

<i>x_pos</i>	X coordinate of top left corner
<i>y_pos</i>	Y coordinate of top left corner
<i>length</i>	The length of the parallelogram
<i>width</i>	The width of the parallelogram
<i>angle</i>	The angle of the bottom corner
<i>colour</i>	Colour of the parallelogram

## Return values

<i>Error</i>	
--------------	--

## Author

Osman Pekcan

### 7.15.2.7 drawPixel()

```
__weak void drawPixel (
    uint16_t xp,
    uint16_t yp,
    uint8_t colour )
```

Weak declared Draw pixel function for if the user wants to implement his own display interface.

By declaring this function in the user his own code the user can implement his own display interface code and still use this graphics library.

## Parameters

<i>x</i>	x coordinate of the pixel
<i>y</i>	y coordinate of the pixel
<i>color</i>	color of the pixel

## Author

Tjerk ten Dam

### 7.15.2.8 drawRect()

```
int drawRect (
    uint16_t x_pos,
```

```

uint16_t y_pos,
uint16_t length,
uint16_t width,
uint8_t colour,
uint8_t filled,
uint8_t rectborder,
uint8_t border_colour )

```

This function draws a rectangle either filled or not.

This function draw either a filled or unfilled rectangle, using filled parameter to decide which one. If it filled, its loops from y axis, drawing lines (using mylijntekenaar) from x axis. If it unfilled draws four lines of the unfilled rectangle

#### Parameters

<i>x_pos,y_pos</i>	Upper left position of the rectangle.
<i>length,width</i>	Length and width of the rectangle.
<i>colour</i>	colour of rectangle.
<i>filled</i>	Parameter that determines whether the rectangle is filled or not
<i>rectborder</i>	Parameter that determines if the function draws a border around the filled rectangle.
<i>border_colour</i>	The color value of the border.

#### Return values

<i>Error</i>	
--------------	--

#### Author

Osman Pekcan

### 7.15.2.9 drawText()

```

int drawText (
    int x,
    int y,
    uint8_t colour,
    char tekst[],
    char fontname[],
    uint8_t fontsize,
    uint8_t fontstyle,
    uint8_t spaces,
    uint8_t linefeed )

```

Draw text function, this function prints the text on the screen.

This function prints the text on the screen.

This function prints/draws the text on the screen. The user can choose between Arial= 0 en Consolas=1. Each fonts have 3 styles: Standard, Italic and Bold Each fonts have 2 sizes: 8, 32 Fontstyle Standard=0, Italic=1 and Bold=2. Fontsize 8=0, 32=1. Add more fonts: Step 1: Import the header file into fonts folder. Step 2: Add all the fonts into **fonts[]**. Step 3: Add fontnames into **font\_name[][]**. Step 4: Change the Define FONT\_NUMBER

## Parameters

<i>x,y</i>	Is the position of the starting point of the first letter.
<i>colour</i>	Is the fonts colour.
<i>tekst[]</i>	Is an array with text.
<i>fontname</i>	Can be chosen from fonts of Arial and Consolas.
<i>fontsize</i>	is the size of the font. The size can be chosen from 8 or 32. 0 = 8 and 1 = 32.
<i>frontstyle</i>	is the style of fonts, this can be chosen of Italic, Bold.

## Return values

<i>Error</i>	
--------------	--

## Author

Djalil &amp; Tjerk

## 7.15.2.10 myLijntekenaar()

```
int myLijntekenaar (
    uint16_t x_begin,
    uint16_t y_begin,
    uint16_t x_eind,
    uint16_t y_eind,
    uint8_t colour )
```

this function draws a single line from a to b

This function starts with 5 variables, x and y coordinate for the begin and x and y coordinates for the end. The last variable as colour of the line. The function first calculates the offset of both x and y coordinates. When either offsets is equal to zero, the line becomes straight. If y offset is zero the function draws horizontal line, else a vertical line. In the situation that neither offset are zero, the function calculates the derivative of the line. The derivative of the line determines the angle that the line has. A derivative value lower than one has a small angle. If the angle is *small* (derivative < 1) the y coordinates is calculated with the derivative. If the angle is *big* (derivative > 1) it is the inverse of the calculation of the smaller angle. After the function looks whether the derivative is higher than one, the function looks if the derivative is positive or negative. This determines if some parameter is positive or negative. Lastly the function looks at orientation of the line with respect to the x axis, which decides which parameter is the start point and end point of the calculation.

## Parameters

<i>x_begin, y_begin</i>	Coordinates where the line starts.
<i>x_eind, y_eind</i>	Coordinates where the line ends.
<i>colour</i>	colour of the line

## Return values

<i>Error</i>	
--------------	--

**Author**

Osman Pekcan

**Note**

Possible to optimize this function, since there was a finite amount of time this project needed to handed in.

**Warning**

It is possible to relocate the derivative calculation but when the x offset is zero, the calculate would have a division by zero.

Here is the caller graph for this function:



## 7.16 Core/Src/front\_layer.c File Reference

```
#include "front_layer.h"
#include "usart.h"
#include "error.h"
#include "logic_layer.h"
```

### Functions

- void `FL_UART_Init` ()  
*Function for initialisatie of the UART.*
- void `softonErrorHandler` (uint8\_t error)
- void `error_on_screen` (uint8\_t error)

### Variables

- `input_vars` input

#### 7.16.1 Detailed Description

**Author**

Tjerk ten Dam

**Version**

1.0

## 7.16.2 Function Documentation

### 7.16.2.1 FL\_UART\_Init()

```
void FL_UART_Init ( )
```

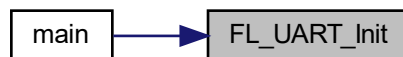
Function for initialisatie of the UART.

First the function empties out the buffer and initialist the HAL UART

Author

Tjerk ten Dam

Here is the caller graph for this function:



## 7.17 Core/Src/logic\_layer.c File Reference

: Logic layer

```

#include "logic_layer.h"
#include "stm32_ub_vga_screen.h"
#include "error.h"
#include "tim.h"
#include <string.h>

```

### Functions

- `uint8_t font_style` (char command[])  
*This function translates font\_style text to variable.*
- `uint8_t r3g3b2_Colour` (char command[])  
*This function is the 8 bit color designation 8bit color (R3G3B2) Red (3bit) -> Bit7-Bit5 Green (3bit) -> Bit4-Bit2 Blue (2bit) -> Bit1-Bit0.*
- `void buffer_init` ()  
*This function is for initilising buffer of scripts.*
- `void HAL_TIM_PeriodElapsedCallback` (TIM\_HandleTypeDef \*htim)  
*Tim callback when period is elapsed.*
- `void writeBuffer` (char buf[])  
*This function writes a array to the circular script buffer.*
- `int indexBuffer` (int index)  
*This function translates an index for the script buffer.*
- `int logic_layer` (char commando[])  
*This function looks at input array and calls function the input array calls for.*

## Variables

- [wait\\_vars](#) wait

### 7.17.1 Detailed Description

: Logic layer

### 7.17.2 Function Documentation

#### 7.17.2.1 `buffer_init()`

```
void buffer_init ( )
```

This function is for initilising buffer of scripts.

#### Authors

Tjerk ten Dam

#### 7.17.2.2 `font_style()`

```
uint8_t font_style (
    char command[ ] )
```

This function translates font\_style text to variable.

#### Parameters

<i>command</i>	This is the font_style string.
----------------	--------------------------------

#### Return values

<i>Variable</i>	for tekst function
<i>0</i>	= normal tekst
<i>1</i>	= bold tekst
<i>2</i>	= Italic tekst

#### Authors

Osman Pekcan,Tjerk ten Dam



### 7.17.2.3 HAL\_TIM\_PeriodElapsedCallback()

```
void HAL_TIM_PeriodElapsedCallback (
    TIM_HandleTypeDef * htim )
```

Tim callback when period is elapsed.

This callback functions gets called when the timer period is elapsed for the wait function and when elapsed the wait flag gets reset.

#### Parameters

<i>*htim</i>	handle of the timer
--------------	---------------------

#### Authors

Tjerk ten Dam

### 7.17.2.4 indexBuffer()

```
int indexBuffer (
    int index )
```

This function translates an index for the script buffer.

#### Parameters

<i>index</i>	index for the script buffer
--------------	-----------------------------

#### Return values

<i>correct</i>	index for the circular buffer
----------------	-------------------------------

#### Authors

Tjerk ten Dam

### 7.17.2.5 logic\_layer()

```
int logic_layer (
    char commando[] )
```

This function looks at input array and calls function the input array calls for.

First script commando's are split up and put into array: `commando_filled`. Then the function looks which script is called. If the function found the script that is called, the corresponding function is called.

**Parameters**

<i>commando</i>	This is the script commando.
-----------------	------------------------------

**Authors**

Osman Pekcan,Tjerk ten Dam

**7.17.2.6 r3g3b2\_Colour()**

```
uint8_t r3g3b2_Colour (
    char command[ ] )
```

This function is the 8 bit color designation 8bit color (R3G3B2) Red (3bit) -> Bit7-Bit5 Green (3bit) -> Bit4-Bit2 Blue (2bit) -> Bit1-Bit0.

**Parameters**

<i>commando[ ]</i>	this is the
--------------------	-------------

**Authors**

Djalil

**7.17.2.7 writeBuffer()**

```
void writeBuffer (
    char buf[ ] )
```

This function writes a array to the circular script buffer.

**Parameters**

<i>buf</i>	array to be write to the script buffer
------------	--

**Authors**

Tjerk ten Dam

**7.18 Core/Src/main.c File Reference**

: Main program body

```
#include "main.h"
#include "dma.h"
#include "tim.h"
#include "usart.h"
#include "gpio.h"
#include "logic_layer.h"
#include "front_layer.h"
```

## Macros

- `#define USART_PRINTF` int fputc(int ch, FILE \*f)

## Functions

- void [SystemClock\\_Config](#) (void)  
*System Clock Configuration.*
- int [main](#) (void)  
*The application entry point.*
- void [Error\\_Handler](#) (void)  
*This function is executed in case of error occurrence.*

## Variables

- volatile char **container** [1024]
- volatile int **temp**
- volatile int **key**
- [USART\\_PRINTF](#)
- return **ch**

### 7.18.1 Detailed Description

: Main program body

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: [opensource.org/licenses/BSD-3-Clause](https://opensource.org/licenses/BSD-3-Clause)

### 7.18.2 Function Documentation

#### 7.18.2.1 Error\_Handler()

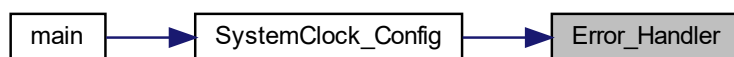
```
void Error_Handler (  
    void )
```

This function is executed in case of error occurrence.

## Return values

<i>None</i>	
-------------	--

Here is the caller graph for this function:



### 7.18.2.2 main()

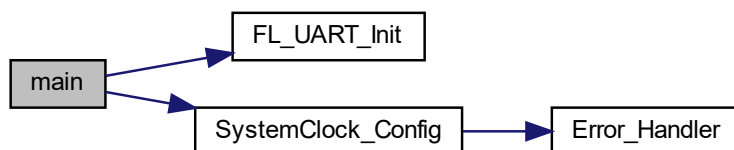
```
int main (  
    void )
```

The application entry point.

## Return values

<i>int</i>	
------------	--

Here is the call graph for this function:



### 7.18.2.3 SystemClock\_Config()

```
void SystemClock_Config (  
    void )
```

System Clock Configuration.

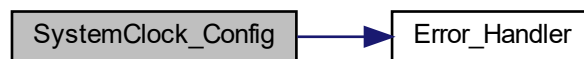
## Return values

<i>None</i>	
-------------	--

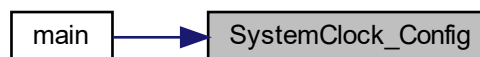
Configure the main internal regulator output voltage

Initializes the RCC Oscillators according to the specified parameters in the RCC\_OscInitTypeDef structure.

Initializes the CPU, AHB and APB buses clocksHere is the call graph for this function:



Here is the caller graph for this function:



### 7.18.3 Variable Documentation

#### 7.18.3.1 USART\_PRINTF

USART\_PRINTF

**Initial value:**

```
{
    HAL_UART_Transmit(&huart2, (uint8_t *)&ch, 1, 0xFFFF)
```

## 7.19 Core/Src/stm32f4xx\_it.c File Reference

Interrupt Service Routines.

```
#include "main.h"
#include "stm32f4xx_it.h"
#include "front_layer.h"
#include "logic_layer.h"
```

## Functions

- void **NMI\_Handler** (void)  
*This function handles Non maskable interrupt.*
- void **HardFault\_Handler** (void)  
*This function handles Hard fault interrupt.*
- void **MemManage\_Handler** (void)  
*This function handles Memory management fault.*
- void **BusFault\_Handler** (void)  
*This function handles Pre-fetch fault, memory access fault.*
- void **UsageFault\_Handler** (void)  
*This function handles Undefined instruction or illegal state.*
- void **SVC\_Handler** (void)  
*This function handles System service call via SWI instruction.*
- void **DebugMon\_Handler** (void)  
*This function handles Debug monitor.*
- void **PendSV\_Handler** (void)  
*This function handles Pendable request for system service.*
- void **SysTick\_Handler** (void)  
*This function handles System tick timer.*
- void **TIM2\_IRQHandler** (void)  
*This function handles TIM2 global interrupt.*
- void **USART2\_IRQHandler** (void)  
*This function handles USART2 global interrupt.*
- void **TIM5\_IRQHandler** (void)  
*This function handles TIM5 global interrupt.*
- void **DMA2\_Stream5\_IRQHandler** (void)  
*This function handles DMA2 stream5 global interrupt.*

## Variables

- DMA\_HandleTypeDef **hdma\_tim1\_up**
- TIM\_HandleTypeDef **htim2**
- TIM\_HandleTypeDef **htim5**
- UART\_HandleTypeDef **huart2**
- TIM\_HandleTypeDef **htim1**

### 7.19.1 Detailed Description

Interrupt Service Routines.

Attention

© Copyright (c) 2020 STMicroelectronics. All rights reserved.

This software component is licensed by ST under BSD 3-Clause license, the "License"; You may not use this file except in compliance with the License. You may obtain a copy of the License at: [opensource.org/licenses/BSD-3-Clause](https://opensource.org/licenses/BSD-3-Clause)

# Index

API.c  
  drawBitmap, [217](#)  
  drawCircle, [217](#)  
  drawCircleplus, [218](#)  
  drawFigure, [219](#)  
  drawLines, [219](#)  
  drawParallelogram, [220](#)  
  drawPixel, [221](#)  
  drawRect, [221](#)  
  drawText, [222](#)  
  myLijntekenaar, [223](#)

API.h  
  drawBitmap, [18](#)  
  drawCircle, [19](#)  
  drawCircleplus, [20](#)  
  drawFigure, [21](#)  
  drawLines, [22](#)  
  drawParallelogram, [23](#)  
  drawPixel, [23](#)  
  drawRect, [24](#)  
  drawText, [25](#)  
  myLijntekenaar, [26](#)

bitmap.h  
  smiley\_map, [29](#)

buffer\_init  
  logic\_layer.c, [226](#)  
  logic\_layer.h, [208](#)

Core/Inc/API.h, [17](#), [27](#)  
Core/Inc/bitmap.h, [28](#), [29](#)  
Core/Inc/error.h, [204](#), [205](#)  
Core/Inc/front\_layer.h, [206](#), [207](#)  
Core/Inc/logic\_layer.h, [207](#), [210](#)  
Core/Inc/main.h, [211](#), [213](#)  
Core/Inc/stm32f4xx\_it.h, [214](#), [215](#)  
Core/Src/API.c, [216](#)  
Core/Src/front\_layer.c, [224](#)  
Core/Src/logic\_layer.c, [225](#)  
Core/Src/main.c, [228](#)  
Core/Src/stm32f4xx\_it.c, [231](#)

drawBitmap  
  API.c, [217](#)  
  API.h, [18](#)

drawCircle  
  API.c, [217](#)  
  API.h, [19](#)

drawCircleplus  
  API.c, [218](#)

  API.h, [20](#)

drawFigure  
  API.c, [219](#)  
  API.h, [21](#)

drawLines  
  API.c, [219](#)  
  API.h, [22](#)

drawParallelogram  
  API.c, [220](#)  
  API.h, [23](#)

drawPixel  
  API.c, [221](#)  
  API.h, [23](#)

drawRect  
  API.c, [221](#)  
  API.h, [24](#)

drawText  
  API.c, [222](#)  
  API.h, [25](#)

Error\_Handler  
  main.c, [229](#)  
  main.h, [212](#)

FL\_UART\_Init  
  front\_layer.c, [225](#)  
  front\_layer.h, [206](#)

font\_style  
  logic\_layer.c, [226](#)

front\_layer.c  
  FL\_UART\_Init, [225](#)

front\_layer.h  
  FL\_UART\_Init, [206](#)

HAL\_TIM\_PeriodElapsedCallback  
  logic\_layer.c, [226](#)  
  logic\_layer.h, [209](#)

indexBuffer  
  logic\_layer.c, [227](#)  
  logic\_layer.h, [209](#)

input\_vars, [15](#)

logic\_layer  
  logic\_layer.c, [227](#)  
  logic\_layer.h, [210](#)

logic\_layer.c  
  buffer\_init, [226](#)  
  font\_style, [226](#)  
  HAL\_TIM\_PeriodElapsedCallback, [226](#)  
  indexBuffer, [227](#)

- logic\_layer, [227](#)
- r3g3b2\_Colour, [228](#)
- writeBuffer, [228](#)
- logic\_layer.h
  - buffer\_init, [208](#)
  - HAL\_TIM\_PeriodElapsedCallback, [209](#)
  - indexBuffer, [209](#)
  - logic\_layer, [210](#)
  - writeBuffer, [210](#)
- main
  - main.c, [230](#)
- main.c
  - Error\_Handler, [229](#)
  - main, [230](#)
  - SystemClock\_Config, [230](#)
  - USART\_PRINTF, [231](#)
- main.h
  - Error\_Handler, [212](#)
- myLijntekenaar
  - API.c, [223](#)
  - API.h, [26](#)
- r3g3b2\_Colour
  - logic\_layer.c, [228](#)
- smiley\_map
  - bitmap.h, [29](#)
- SystemClock\_Config
  - main.c, [230](#)
- USART\_PRINTF
  - main.c, [231](#)
- wait\_vars, [15](#)
- writeBuffer
  - logic\_layer.c, [228](#)
  - logic\_layer.h, [210](#)