Software Ontwikkeling

Generated by Doxygen 1.10.0

Chapter 1

23.24-D-Softwareontwikkeling

jaar 23/24 periode D Software Ontwikkeling

1.1 Project Description

The demo application follows the 3-Tier model and showcase the EE-API-LIB functions in the third layer. It will execute commands from a scripting language with the following structure:

Front Layer: Reads script commands. Logic Layer: Contains drawing functionality. I/O Layer: Manages input and output to the hardware. By completing this project, We will deliver a fully operational solution with the EE-API-library and a client application, along with comprehensive documentation and professional source code. This will enable IP's in-house programmers to gain practical experience with the new VGA screens.

1.1.1 Team members

- Michel Vollmuller
- Tim Wannet
- Tijmen Willems

1.1.2 Project workflow

- **Development Tools**: Visual Studio Code, Git, GitHub Desktop and Gitkraken.
- **Methodology**: Scrum with Github as scrum board. (https://github.com/users/Ragazzoforte/projects/2/views/1)
- **Communication Tools**: WhatsApp and GitHub (scrum).
- **Documentation Tools**: Doxygen and GitHub. (https://ragazzoforte.github.io/23.24-D-Softwareontwikkeling/)
- **Version Control**: GitHub
- **Coding Standards**: C++ Coding Standards

1.2 Project rules

1.2.1 Branches

From every issue, a new branch will be created. The branch name will be the same as the issue name. The branch will be merged into the main branch after the issue is completed. Every branch will be deleted after merging.

A Branch name should always start with the following prefixes:

- Feature Branches: feature/.
- Bugfix Branches: bugfix/.
- Hotfix Branches: hotfix/.
- Release Branches: release/.
- Documentation Branches: docs/.

1.3 Manual

1.3.1 3-layer model

The 3-layer model, also known as the layered architecture, is a software design pattern that separates the functionality of a program into three distinct layers: the user interface (UI) layer, the logic layer, and the I/O (input/output) layer. Each layer has a specific responsibility and interacts with the other layers in a controlled manner.

1. User Interface (UI) Layer: The User Interface Layer the topmost layer of the application and is the only layer that interacts directly with the user. The UI Layer is responsible for displaying the user interface, handling user input, and presenting the output of the program to the user. It is also responsible for translating user input into commands that can be understood by the logic layer.
2. Logic Layer: The logic layer contains the core functionality and business logic of the program. Here the commands are translated into their respective functions. The layer can utilize text, fonts and bitmaps to draw on the screen.
3. I/O (Input/Output) Layer: The I/O Layer handles the communication between the program and the user. It is responsible for reading input data from a command line and writing data to the command line. This layer abstracts the details of the underlying I/O operations, providing a consistent interface for the logic layer to interact with different data sources. Input and output data is being handled by the SerialPort_Terminal.exe program that connect via serial to the STM32F407VG microcontroller. The layer also handles the communication between the program and the VGA screen to display the output. The VGA screen is connected to the microcontroller via a VGA cable. To optimise modularity the i/o layer is seperated in 2 sections: the UART driver and the VGA driver.

By separating the concerns into different layers. The 3-layer model promotes modularity, reusability, and maintainability of the code. It allows for easier testing, as each layer can be tested independently. It provides a clear separation of responsibilities, making the code easier to understand and modify.

1.3.2 Graphical Design

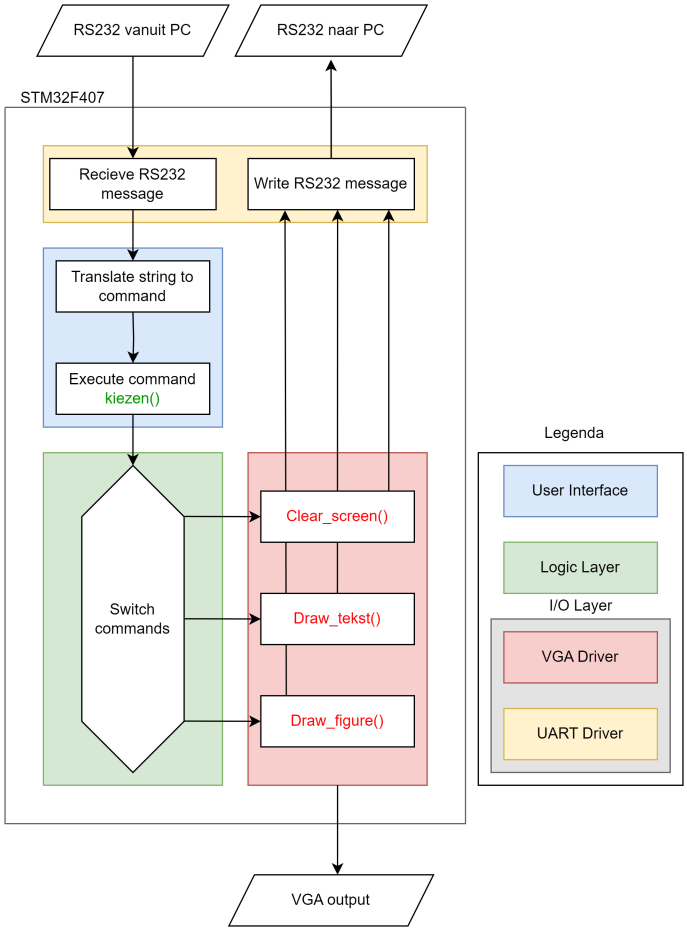

Figure 1.1 Local Image

From the image above you can see the 3-layer model representeted in a high level design. As explained in the last chapter the 3-layer model is separated in 4 different parts: User Interface, Logic Layer, VGA Driver and UART Driver.

1. User Interface The User Interface (blue) is represented by 2 blocks in the HLD: "Translat_string_to_command" and "Execute_command: kiezen()". It recieves a string from the terminal by the UART Driver, translates this string to a command and calles the fuction defined in the Logic Layer.

The User Interface is being handled by the following files:

- user_interface.c/h: mostly handles the incoming string and translates this to the respected function.

1. Logic Layer Inside the Logic Layer (green) the incoming command from the User Interface. Here the command is being translated to the correct function that is defined in the VGA Driver. The Logic Layer also defines the fonts and bitmaps that are being used in the program.

The Logic Later is being handled by the following files:

- Logic_layer.c/h: Chooses the correct function based on the incoming message (from uart.c/h)

- font.c/h: Contains all the fonts that are being used in the program.
- bitmap.c/h: Contains all the bitmaps that are being used in the program.

1. VGA Driver The VGA Driver (red) is the layer that is responsible for drawing on the screen. Errors are being handled in the driver and send to the UART Driver to be displayed on the terminal. The switch case from the Logic Layer refers to 3 functions in the VGA Driver, these are example functions and represent all the functions that are defined in the VGA Driver.

The VGA Driver is being handled by the following files:

- vga_driver.c/h: Has all functions that are being used to draw on the screen or are being used to help the functions.

1. UART Driver This Layer (yellow) is used for the communication between the terminal and the User Interface. It recieves the incoming string from terminal and sends the errors back received from the VGA Driver.

The UART Driver is being handled by the following files:

- uart.c/h: With the lack of HAL libraries there was the need of making a own UART program so the microcontroller could communicate with the terminal. This program is optimized to minimize stuttering, minimize sync issues and optimize speed.

1.3.3 Scipts-commands

- lijn, x, y, x', y', kleur, dikte
- rechthoek, x_lup, y_lup, breedte, hoogte, kleur, gevuld (1,0) [als 1: rand (1px) met kleur]
- tekst, x, y, kleur, tekst, fontnaam (arial, consolas), fontgrootte (1,2), fontstijl (normaal, vet, cursief)
- bitmap, nr, x-lup, y-lup [tenminste: pijl (in 4 richtingen), smiley (boos, blij)]
- clearscherm, kleur
- polygon, x, y, size, corner, kleur, reserved
- cirkel, x, y, radius, kleur, gevuld (1,0) [als 1: rand (1px) met kleur]

1.3.4 Colours

zwart, blauw, lichtblauw, groen, lichtgroen, cyaan, lichtcyaan, rood, lichtrood, magenta, lichtmagenta, bruin, geel, grijs, wit

Chapter 2

Topic Index

2.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

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

Chapter 5

Topic Documentation

5.1 driver functions

Macros

- #define **ZWART** 540422306
- #define **LICHTMAGENTA** 338820699
- #define **MAGENTA** 3940791655
- #define **BLAUW** 511564997
- #define **LICHTBLAUW** 1778211001
- #define **CYAAN** 513217430
- #define **LICHTCYAAN** 1779863434
- #define **GROEN** 517724933
- #define **LICHTGROEN** 1784370937
- #define **GEEL** 15674151
- #define **ROOD** 16080670
- #define **LICHTROOD** 4089130130
- #define **BRUIN** 511801994
- #define **GRIJS** 517718569
- #define **WIT** 492542
- #define **PI** 3.14159265
- #define **TERMS** 4
- #define **UNUSED**(x) (void)(x)

Functions

- void **USART2_IRQHandler** (void)
- void UART_Init (uint32_t baudrate)

  Initializes the UART interface.
- void **UART_SendChar** (char c)
- void UART_SendString (char ∗string)

  Sends a single character over the UART interface. the function blocks untill the character is send.
- int API_draw_text (int x_lup, int y_lup, int color, char ∗text, char ∗fontname, int fontsize, int fontstyle, int reserved)

  Draws a string to the VGA screen.
- int API_draw_line (int x1, int y1, int x2, int y2, int colour, int thickness, int reserved)

Draw a line on the VGA screen.

- int API_draw_rectangle (int x, int y, int width, int height, int colour, int filled, int reserved1, int reserved2)

  API_draw_rectangle() is used to draw a rectangle to the VGA screen.

- int API_draw_polygon (int x, int y, int size, int corners, int colour, int reserved)

  API_draw_polygon() is used to draw a polygon to the VGA screen.

- int API_draw_bitmap (int x_lup, int y_lup, int bm_nr)

  Draws a bitmap to the VGA screen.

- int API_clearscreen (int colour)

  API_clearscreen() is used to clear the VGA screen.

- unsigned long hash (char *str)

  hash() is used to hash a string to a unique value.

- uint8_t color_chooser (char *str)

  gives the corresponding colour value for the given string

Variables

- char **UART_TX_message** [UART_BUFFER_SIZE]
- char **UART_RX_message** [UART_BUFFER_SIZE]
- uint16_t **charCnt** = 0
- bool **msgReceivedUSART2** = false
- const double cos_table [73]
- const double sin_table [73]

5.1.1 Detailed Description

5.1.2 Function Documentation

5.1.2.1 API_clearscreen()

```
int API_clearscreen (
            int colour )
```

API_clearscreen() is used to clear the VGA screen.

Note

selected color must be predefined, valid colours: zwart, lichtmagenta, magenta, blauw, lichtblauw, cyaan, lichtcyaan groen, lichtgroen, geel, rood, lichtrood, bruin, grijs, wit

Parameters

colour	the color to clear the screen with

Return values

none	

5.1.2.2 API_draw_bitmap()

```
int API_draw_bitmap (
            int x_lup,
            int y_lup,
            int bm_nr )
```

Draws a bitmap to the VGA screen.

This function draws a bitmap to the VGA screen at the specified coordinates. The bitmap is selected by number from a predefined list of bitmaps.

Parameters

x_lup	The x-coordinate of the left upper point where the bitmap should be drawn.
y_lup	The y-coordinate of the left upper point where the bitmap should be drawn.
bm_nr	The number of the bitmap to be drawn. This corresponds to an index in the predefined list of bitmaps. The following bitmaps are available: • 1: Smiley happy • 2: Smiley sad • 3: Arrow up • 4: Arrow right • 5: Arrow down • 6: Arrow left • 7: Megaman

Returns

Returns 0 on success, non-zero error code on failure.

5.1.2.3 API_draw_line()

```
int API_draw_line (
            int x1,
            int y1,
            int x2,
            int y2,
            int colour,
            int thickness,
            int reserved )
```

Draw a line on the VGA screen.

Note

This function uses the Bresenham's line algorithm to draw a line on the VGA screen.

Parameters

x1	the x-coordinate of the starting point of the line
y1	the y-coordinate of the starting point of the line
x2	the x-coordinate of the ending point of the line
y2	the y-coordinate of the ending point of the line
colour	the colour of the line
weight	the weight/thicknes of the line
reserved	reserved for future use

Return values

none	

5.1.2.4 API_draw_polygon()

```
int API_draw_polygon (
            int x,
            int y,
            int size,
            int corners,
            int colour,
            int reserved )
```

API_draw_polygon() is used to draw a polygon to the VGA screen.

Note

selected polygon must not exceed a certain size

Parameters

x	the x-coordinate where the polygon should start drawing
y	the y-coordinate where the polygon should start drawing
size	the size of the polygon
corners	the number of corners the polygon should have
colour	the colour of the polygon

Return values

none	

5.1.2.5 API_draw_rectangle()

```
int API_draw_rectangle (
            int x,
            int y,
            int width,
            int height,
            int colour,
            int filled,
            int reserved1,
            int reserved2 )
```

API_draw_rectangle() is used to draw a rectangle to the VGA screen.

Note

To draw a rotated rectangle use 'API_draw_line()'

Parameters

x	the x-coordinate of the top-left corner of the rectangle
y	the y-coordinate of the top-left corner of the rectangle
width	the width of the rectangle
height	the height of the rectangle
colour	the colour of the rectangle
filled	whether the rectangle should be filled in or not
reserved1	reserved for future use
reserved2	reserved for future use

Return values

5.1.2.6 API_draw_text()

```
int API_draw_text (
            int x_lup,
            int y_lup,
            int color,
            char * text,
            char * fontname,
            int fontsize,
            int fontstyle,
            int reserved )
```

Draws a string to the VGA screen.

This function draws a string to the VGA screen using the specified font, size, and style.

Parameters

x_lup	The x-coordinate of the left upper point where the text should start.
y_lup	The y-coordinate of the left upper point where the text should start.
color	The color of the text.
text	The text to be drawn.
fontname	The name of the font to be used. arial or consolas
fontsize	The size of the font. 1 for small, 2 for big.
fontstyle	The style of the font. Use the predefined constants for this. 1 for normal, 2 for italic, 3 for bold.
reserved	Reserved for future use.

Returns

Returns 0 on success, non-zero error code on failure.

5.1.2.7 color_chooser()

```
uint8_t color_chooser (
            char * str )
```

gives the corresponding colour value for the given string

Parameters

str	The string to be converted to a colour value

Returns

Returns the color value on success.

Returns 100 on error.

5.1.2.8 hash()

```
unsigned long hash (
            char * str )
```

hash() is used to hash a string to a unique value.

Note

This function is used to generate a unique value for the string.

Parameters

str	the string to be hashed

Return values

unsigned	long: the hashed value of the string

5.1.2.9 UART_Init()

```
void UART_Init (
            uint32_t baudrate )
```

Initializes the UART interface.

Parameters

boudrate	set the transmision boudrate, common values are: • 1200 • 2400 • 4800 • 9600 • 19200 • 38400 • 57600 • 115200

Return values

None	

5.1.2.10 UART_SendString()

```
void UART_SendString (
            char * string )
```

Sends a single character over the UART interface. the function blocks untill the character is send.

Parameters

string	any ascii character to be sent

Return values

None	

5.1.3 Variable Documentation

5.1.3.1 cos_table

```
const double cos_table[73]
```

Initial value:

```
= {
    1.0000000, 0.9961947, 0.9848078, 0.9659258, 0.9396926,
    0.9063078, 0.8660254, 0.8191520, 0.7660444, 0.7071068,
    0.6427876, 0.5735764, 0.5000000, 0.4226183, 0.3420201,
    0.2588190, 0.1736482, 0.0871557, 0.0000000, -0.0871557,
    -0.1736482, -0.2588190, -0.3420201, -0.4226183, -0.5000000,
    -0.5735764, -0.6427876, -0.7071068, -0.7660444, -0.8191520,
    -0.8660254, -0.9063078, -0.9396926, -0.9659258, -0.9848078,
    -0.9961947, -1.0000000, -0.9961947, -0.9848078, -0.9659258,
    -0.9396926, -0.9063078, -0.8660254, -0.8191520, -0.7660444,
    -0.7071068, -0.6427876, -0.5735764, -0.5000000, -0.4226183,
    -0.3420201, -0.2588190, -0.1736482, -0.0871557, -0.0000000,
    0.0871557, 0.1736482, 0.2588190, 0.3420201, 0.4226183,
    0.5000000, 0.5735764, 0.6427876, 0.7071068, 0.7660444,
    0.8191520, 0.8660254, 0.9063078, 0.9396926, 0.9659258,
    0.9848078, 0.9961947, 1.0000000, 0.9961947, 0.9848078,
    0.9659258, 0.9396926, 0.9063078, 0.8660254, 0.8191520,
    0.7660444, 0.7071068, 0.6427876, 0.5735764, 0.5000000
}
```

5.1.3.2 sin_table

```
const double sin_table[73]
```

Initial value:

```
= {
    0.0000000, 0.0871557, 0.1736482, 0.2588190, 0.3420201,
    0.4226183, 0.5000000, 0.5735764, 0.6427876, 0.7071068,
    0.7660444, 0.8191520, 0.8660254, 0.9063078, 0.9396926,
    0.9659258, 0.9848078, 0.9961947, 1.0000000, 0.9961947,
    0.9848078, 0.9659258, 0.9396926, 0.9063078, 0.8660254,
    0.8191520, 0.7660444, 0.7071068, 0.6427876, 0.5735764,
    0.5000000, 0.4226183, 0.3420201, 0.2588190, 0.1736482,
    0.0871557, 0.0000000, -0.0871557, -0.1736482, -0.2588190,
    -0.3420201, -0.4226183, -0.5000000, -0.5735764, -0.6427876,
    -0.7071068, -0.7660444, -0.8191520, -0.8660254, -0.9063078,
    -0.9396926, -0.9659258, -0.9848078, -0.9961947, -1.0000000,
    -0.9961947, -0.9848078, -0.9659258, -0.9396926, -0.9063078,
    -0.8660254, -0.8191520, -0.7660444, -0.7071068, -0.6427876,
    -0.5735764, -0.5000000, -0.4226183, -0.3420201, -0.2588190,
    -0.1736482, -0.0871557, -0.0000000, 0.0871557, 0.1736482,
    0.2588190, 0.3420201, 0.4226183, 0.5000000, 0.5735764,
    0.6427876, 0.7071068, 0.7660444, 0.8191520, 0.8660254,
    0.9063078, 0.9396926, 0.9659258, 0.9848078, 0.9961947,
    1.0000000
}
```

Chapter 6

Data Structure Documentation

6.1 command Struct Reference

Data Fields

- char * **arg** [20]

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

- VGA_Driver/Core/Inc/user_interface.h

6.2 VGA_t Struct Reference

Data Fields

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

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

- VGA_Driver/Core/Inc/stm32_ub_vga_screen.h

Chapter 7

File Documentation

7.1 VGA_Driver/Core/Inc/bitmap.h File Reference

headerfile of bitmap.c

Macros

- #define **MEGAMAN_WIDTH** 21
- #define **MEGAMAN_HEIGHT** 24
- #define **SMILEY_WIDTH** 40
- #define **SMILEY_HEIGHT** 40
- #define **ARROW_UP_WIDTH** 27
- #define **ARROW_UP_HEIGHT** 41
- #define **ARROW_DOWN_WIDTH** 27
- #define **ARROW_DOWN_HEIGHT** 41
- #define **ARROW_LEFT_WIDTH** 41
- #define **ARROW_LEFT_HEIGHT** 27
- #define **ARROW_RIGHT_WIDTH** 41
- #define **ARROW_RIGHT_HEIGHT** 27
- #define **MICHIEL_WIDTH** 320
- #define **MICHIEL_HEIGHT** 213
- #define **FRANC_WIDTH** 240
- #define **FRANC_HEIGHT** 240
- #define **GROEP_WIDTH** 320
- #define **GROEP_HEIGHT** 200

Variables

- const uint8_t megaman [ ]

  Bitmap data for the "Megaman" sprite.
- const uint8_t megaman_2 [MEGAMAN_WIDTH ∗MEGAMAN_HEIGHT]

  A bitmap representation of an megaman image.
- const uint8_t smiley_happy [SMILEY_WIDTH ∗SMILEY_HEIGHT]

  A bitmap representation of a happy smiley image.
- const uint8_t smiley_sad [SMILEY_WIDTH ∗SMILEY_HEIGHT]

  A bitmap representation of a sad smiley image.

- const uint8_t arrow_up [ARROW_UP_WIDTH ∗ARROW_UP_HEIGHT]

  A bitmap representation of a upward arrow image.
- const uint8_t arrow_down [ARROW_DOWN_WIDTH ∗ARROW_DOWN_HEIGHT]

  A bitmap representation of a downward arrow image.
- const uint8_t arrow_left [ARROW_LEFT_WIDTH ∗ARROW_LEFT_HEIGHT]

  A bitmap representation of a left arrow image.
- const uint8_t arrow_right [ARROW_RIGHT_WIDTH ∗ARROW_RIGHT_HEIGHT]

  A bitmap representation of a right arrow image.
- const uint8_t michiel [MICHIEL_WIDTH ∗MICHIEL_HEIGHT]

  A bitmap representation of Michiel Scager.
- const uint8_t franc [FRANC_WIDTH ∗FRANC_HEIGHT]

  A bitmap representation of Franc.
- const uint8_t groep [GROEP_WIDTH ∗GROEP_HEIGHT]

  A bitmap representation of our group image.

7.1.1 Detailed Description

headerfile of bitmap.c

Author

Michel Vollmuller (michel.vollmuller@student.hu.nl)

Tim Wannet (tim.wannet@student.hu.nl)

Tijmen Willems (tijmen.willems@student.hu.nl)

Version

0.1

Date

05-06-2024

Copyright

Copyright (c) 2024

7.1.2 Variable Documentation

7.1.2.1 arrow_down

```
const uint8_t arrow_down[ARROW_DOWN_WIDTH *ARROW_DOWN_HEIGHT]  [extern]
```

A bitmap representation of a downward arrow image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants ARROW_DOWN_WIDTH and ARROW_DOWN_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.1.2.2 arrow_left

```
const uint8_t arrow_left[ARROW_LEFT_WIDTH *ARROW_LEFT_HEIGHT]  [extern]
```

A bitmap representation of a left arrow image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants ARROW_LEFT_WIDTH and ARROW_LEFT_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.1.2.3 arrow_right

```
const uint8_t arrow_right[ARROW_RIGHT_WIDTH *ARROW_RIGHT_HEIGHT]  [extern]
```

A bitmap representation of a right arrow image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants ARROW_RIGHT_WIDTH and ARROW_RIGHT_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.1.2.4 arrow_up

```
const uint8_t arrow_up[ARROW_UP_WIDTH *ARROW_UP_HEIGHT]  [extern]
```

A bitmap representation of a upward arrow image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants ARROW_UP_WIDTH and ARROW_UP_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.1.2.5 franc

```
const uint8_t franc[FRANC_WIDTH *FRANC_HEIGHT]  [extern]
```

A bitmap representation of Franc.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants FRANC_WIDTH and FRANC_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.1.2.6 groep

```
const uint8_t groep[GROEP_WIDTH *GROEP_HEIGHT]  [extern]
```

A bitmap representation of our group image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants GROUP_WIDTH and GROUP_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.1.2.7 megaman

```
const uint8_t megaman[]  [extern]
```

Bitmap data for the "Megaman" sprite.

This array contains the bitmap data for the "Megaman" sprite. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of the sprite. The height and width of the sprite are determined by the size of this array and the specific layout of the bitmap data.

7.1.2.8 megaman_2

```
const uint8_t megaman_2[MEGAMAN_WIDTH *MEGAMAN_HEIGHT]  [extern]
```

A bitmap representation of an megaman image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image is likely to be 21x24 pixels (504 elements) given the size of the array. The color of each pixel is represented as an 8-bit unsigned integer.

7.1.2.9 michiel

```
const uint8_t michiel[MICHIEL_WIDTH *MICHIEL_HEIGHT]  [extern]
```

A bitmap representation of Michiel Scager.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants MICHIEL_WIDTH and MICHIEL_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.1.2.10 smiley_happy

```
const uint8_t smiley_happy[SMILEY_WIDTH *SMILEY_HEIGHT]  [extern]
```

A bitmap representation of a happy smiley image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants SMILEY_WIDTH and SMILEY_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.1.2.11 smiley_sad

```
const uint8_t smiley_sad[SMILEY_WIDTH *SMILEY_HEIGHT]  [extern]
```

A bitmap representation of a sad smiley image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants SMILEY_WIDTH and SMILEY_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.2 bitmap.h

Go to the documentation of this file.

```
00001
00014 #ifndef bitmap_h
00015 #define bitmap_h
00016
00017
00018 #define MEGAMAN_WIDTH  21
00019 #define MEGAMAN_HEIGHT  24
00020
00021 #define SMILEY_WIDTH  40
00022 #define SMILEY_HEIGHT  40
00023
00024 #define ARROW_UP_WIDTH  27
00025 #define ARROW_UP_HEIGHT  41
00026
00027 #define ARROW_DOWN_WIDTH  27
00028 #define ARROW_DOWN_HEIGHT  41
00029
00030 #define ARROW_LEFT_WIDTH  41
00031 #define ARROW_LEFT_HEIGHT  27
00032
00033 #define ARROW_RIGHT_WIDTH  41
00034 #define ARROW_RIGHT_HEIGHT  27
00035
00036 #define MICHIEL_WIDTH  320
00037 #define MICHIEL_HEIGHT  213
00038
00039 #define FRANC_WIDTH  240
00040 #define FRANC_HEIGHT  240
00041
00042 #define GROEP_WIDTH  320
00043 #define GROEP_HEIGHT  200
00044
00045
00046
00047 extern const uint8_t megaman[];
00048 extern const uint8_t megaman_2[MEGAMAN_WIDTH * MEGAMAN_HEIGHT];
00049 extern const uint8_t smiley_happy[SMILEY_WIDTH * SMILEY_HEIGHT];
00050 extern const uint8_t smiley_sad[SMILEY_WIDTH * SMILEY_HEIGHT];
00051 extern const uint8_t arrow_up[ARROW_UP_WIDTH * ARROW_UP_HEIGHT];
00052 extern const uint8_t arrow_down[ARROW_DOWN_WIDTH * ARROW_DOWN_HEIGHT];
00053 extern const uint8_t arrow_left[ARROW_LEFT_WIDTH * ARROW_LEFT_HEIGHT];
00054 extern const uint8_t arrow_right[ARROW_RIGHT_WIDTH * ARROW_RIGHT_HEIGHT];
00055 extern const uint8_t michiel[MICHIEL_WIDTH * MICHIEL_HEIGHT];
00056 extern const uint8_t franc[FRANC_WIDTH * FRANC_HEIGHT];
00057 extern const uint8_t groep[GROEP_WIDTH * GROEP_HEIGHT];
00058
00059
00060 #endif /* bitmap.h */
```

7.3 VGA_Driver/Core/Inc/fonts.h File Reference

headerfile of fonts.c

Macros

- #define **NR_OF_ELEMENTS** 2
- #define **NR_OF_SYMBOLS** 95
- #define **ARIAL_SMALL_HEIGHT** 10
- #define **ARIAL_SMALL_ITALIC_HEIGHT** 12
- #define **ARIAL_SMALL_BOLD_HEIGHT** 11
- #define **ARIAL_LARGE_HEIGHT** 15
- #define **ARIAL_LARGE_ITALIC_HEIGHT** 15
- #define **ARIAL_LARGE_BOLD_HEIGHT** 16
- #define **CONSOLAS_SMALL_HEIGHT** 11
- #define **CONSOLAS_SMALL_ITALIC_HEIGHT** 11
- #define **CONSOLAS_SMALL_BOLD_HEIGHT** 11
- #define **CONSOLAS_LARGE_HEIGHT** 15
- #define **CONSOLAS_LARGE_ITALIC_HEIGHT** 15
- #define **CONSOLAS_LARGE_BOLD_HEIGHT** 15

Variables

- const uint8_t arial_8ptBitmaps [ ]

  Bitmap data for Arial 8pt font.
- const uint8_t arial_italic_8ptBitmaps [ ]

  Bitmap data for Arial 8pt italic font.
- const uint8_t arial_bold_8ptBitmaps [ ]

  Bitmap data for Arial 8pt bold font.
- const uint16_t arial_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 8pt font.
- const uint16_t arial_italic_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 8pt italic font.
- const uint16_t arial_bold_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 8pt bold font.
- const uint8_t arial_11ptBitmaps [ ]

  Bitmap data for Arial 11pt font.
- const uint8_t arial_italic_11ptBitmaps [ ]

  Bitmap data for Arial 11pt italic font.
- const uint8_t arial_bold_11ptBitmaps [ ]

  Bitmap data for Arial 11pt bold font.
- const uint16_t arial_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 11pt font.
- const uint16_t arial_italic_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 11pt italic font.
- const uint16_t arial_bold_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 11pt bold font.
- const uint8_t consolas_8ptBitmaps [ ]

  Bitmap data for Consolas 8pt font.
- const uint8_t consolas_italic_8ptBitmaps [ ]

  Bitmap data for Consolas 8pt italic font.
- const uint8_t consolas_bold_8ptBitmaps [ ]

  Bitmap data for Consolas 8pt bold font.
- const uint16_t consolas_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 8pt font.
- const uint16_t consolas_italic_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 8pt italic font.
- const uint16_t consolas_bold_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 8pt bold font.
- const uint8_t consolas_11ptBitmaps [ ]

  Bitmap data for Consolas 11pt font.
- const uint8_t consolas_italic_11ptBitmaps [ ]

  Bitmap data for Consolas 11pt italic font.
- const uint8_t consolas_bold_11ptBitmaps [ ]

  Bitmap data for Consolas 11pt bold font.
- const uint16_t consolas_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 11pt font.
- const uint16_t consolas_italic_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 11pt italic font.
- const uint16_t consolas_bold_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 11pt bold font.

7.3.1 Detailed Description

headerfile of fonts.c

Author

Michel Vollmuller (michel.vollmuller@gmail.com)
Tim Wannet (tim.wannet@student.hu.nl)
Tijmen Willems (tijmen.willems@student.hu.nl)

Version

0.1

Date

05-06-2024

Copyright

Copyright (c) 2024

7.3.2 Variable Documentation

7.3.2.1 arial_11ptBitmaps

```
const uint8_t arial_11ptBitmaps[]  [extern]
```

Bitmap data for Arial 11pt font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.2 arial_11ptDescriptors

```
const uint16_t arial_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Arial 11pt font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with NR_OF_SYMBOLS rows and NR_OF_ELEMENTS columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.3.2.3 arial_8ptBitmaps

```
const uint8_t arial_8ptBitmaps[]  [extern]
```

Bitmap data for Arial 8pt font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.4 arial_8ptDescriptors

```
const uint16_t arial_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Arial 8pt font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.3.2.5 arial_bold_11ptBitmaps

```
const uint8_t arial_bold_11ptBitmaps[]  [extern]
```

Bitmap data for Arial 11pt bold font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.6 arial_bold_11ptDescriptors

```
const uint16_t arial_bold_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Arial 11pt bold font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.3.2.7 arial_bold_8ptBitmaps

```
const uint8_t arial_bold_8ptBitmaps[]  [extern]
```

Bitmap data for Arial 8pt bold font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.8 arial_bold_8ptDescriptors

```
const uint16_t arial_bold_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Arial 8pt bold font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with NR_OF_SYMBOLS rows and NR_OF_ELEMENTS columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.3.2.9 arial_italic_11ptBitmaps

```
const uint8_t arial_italic_11ptBitmaps[]  [extern]
```

Bitmap data for Arial 11pt italic font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.10 arial_italic_11ptDescriptors

```
const uint16_t arial_italic_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Arial 11pt italic font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with NR_OF_SYMBOLS rows and NR_OF_ELEMENTS columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.3.2.11 arial_italic_8ptBitmaps

```
const uint8_t arial_italic_8ptBitmaps[]  [extern]
```

Bitmap data for Arial 8pt italic font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.12 arial_italic_8ptDescriptors

```
const uint16_t arial_italic_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Arial 8pt italic font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.3.2.13 consolas_11ptBitmaps

```
const uint8_t consolas_11ptBitmaps[]  [extern]
```

Bitmap data for Consolas 11pt font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.14 consolas_11ptDescriptors

```
const uint16_t consolas_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Consolas 11pt font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.3.2.15 consolas_8ptBitmaps

```
const uint8_t consolas_8ptBitmaps[]  [extern]
```

Bitmap data for Consolas 8pt font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.16 consolas_8ptDescriptors

```
const uint16_t consolas_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Consolas 8pt font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with NR_OF_SYMBOLS rows and NR_OF_ELEMENTS columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.3.2.17 consolas_bold_11ptBitmaps

```
const uint8_t consolas_bold_11ptBitmaps[]  [extern]
```

Bitmap data for Consolas 11pt bold font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.18 consolas_bold_11ptDescriptors

```
const uint16_t consolas_bold_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Consolas 11pt bold font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with NR_OF_SYMBOLS rows and NR_OF_ELEMENTS columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.3.2.19 consolas_bold_8ptBitmaps

```
const uint8_t consolas_bold_8ptBitmaps[]  [extern]
```

Bitmap data for Consolas 8pt bold font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.20 consolas_bold_8ptDescriptors

```
const uint16_t consolas_bold_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Consolas 8pt bold font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.3.2.21 consolas_italic_11ptBitmaps

```
const uint8_t consolas_italic_11ptBitmaps[]  [extern]
```

Bitmap data for Consolas 11pt italic font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.22 consolas_italic_11ptDescriptors

```
const uint16_t consolas_italic_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Consolas 11pt italic font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.3.2.23 consolas_italic_8ptBitmaps

```
const uint8_t consolas_italic_8ptBitmaps[]  [extern]
```

Bitmap data for Consolas 8pt italic font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.3.2.24 consolas_italic_8ptDescriptors

```
const uint16_t consolas_italic_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]  [extern]
```

Descriptors for Consolas 8pt italic font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.4 fonts.h

Go to the documentation of this file.

```
00001
00014 #ifndef fonts_h
00015 #define fonts_h
00016
00017 #define NR_OF_ELEMENTS 2
00018 #define NR_OF_SYMBOLS 95
00019
00020 /* Small arial fonts */
00021
00022 #define ARIAL_SMALL_HEIGHT        10
00023 #define ARIAL_SMALL_ITALIC_HEIGHT 12
00024 #define ARIAL_SMALL_BOLD_HEIGHT   11
00025
00026 extern const uint8_t arial_8ptBitmaps[];
00027 extern const uint8_t arial_italic_8ptBitmaps[];
00028 extern const uint8_t arial_bold_8ptBitmaps[];
00029
00030 extern const uint16_t arial_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00031 extern const uint16_t arial_italic_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00032 extern const uint16_t arial_bold_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00033
00034 /* Large arial fonts */
00035
00036 #define ARIAL_LARGE_HEIGHT         15
00037 #define ARIAL_LARGE_ITALIC_HEIGHT  15
00038 #define ARIAL_LARGE_BOLD_HEIGHT    16
00039
00040 extern const uint8_t arial_11ptBitmaps[];
00041 extern const uint8_t arial_italic_11ptBitmaps[];
00042 extern const uint8_t arial_bold_11ptBitmaps[];
00043
```

```
00044 extern const uint16_t arial_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00045 extern const uint16_t arial_italic_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00046 extern const uint16_t arial_bold_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00047
00048
00049
00050 /* Small consolas fonts */
00051
00052 #define CONSOLAS_SMALL_HEIGHT 11
00053 #define CONSOLAS_SMALL_ITALIC_HEIGHT 11
00054 #define CONSOLAS_SMALL_BOLD_HEIGHT 11
00055
00056 extern const uint8_t consolas_8ptBitmaps[];
00057 extern const uint8_t consolas_italic_8ptBitmaps[];
00058 extern const uint8_t consolas_bold_8ptBitmaps[];
00059
00060 extern const uint16_t consolas_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00061 extern const uint16_t consolas_italic_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00062 extern const uint16_t consolas_bold_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00063
00064
00065 /* Large consolas fonts */
00066
00067 #define CONSOLAS_LARGE_HEIGHT 15
00068 #define CONSOLAS_LARGE_ITALIC_HEIGHT 15
00069 #define CONSOLAS_LARGE_BOLD_HEIGHT 15
00070
00071 extern const uint8_t consolas_11ptBitmaps[];
00072 extern const uint8_t consolas_italic_11ptBitmaps[];
00073 extern const uint8_t consolas_bold_11ptBitmaps[];
00074
00075 extern const uint16_t consolas_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00076 extern const uint16_t consolas_italic_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00077 extern const uint16_t consolas_bold_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS];
00078
00079 #endif /* fonts.h */
```

7.5 VGA_Driver/Core/Inc/logic_layer.h File Reference

headerfile of logic_layer.c

```
#include "main.h"
#include "fonts.h"
#include "user_interface.h"
```

Macros

- #define **lijn** 15858359
- #define **rechthoek** 4254663175
- #define **tekst** 532670837
- #define **bitmap** 3993858727
- #define **clearscherm** 96846547
- #define **wacht** 536075777
- #define **herhaal** 1948098847
- #define **cirkel** 4032920196
- #define **figuur** 4149942492
- #define **polygon** 4074764338

Functions

- void kiezen (command str)

  Verwerkt een commando en roept de bijbehorende API-functie aan.

7.5.1 Detailed Description

headerfile of logic_layer.c

Author

Michel Vollmuller (michel.vollmuller@gmail.com)

Tim Wannet (tim.wannet@student.hu.nl)

Tijmen Willems (tijmen.willems@student.hu.nl)

Version

0.1

Date

05-06-2024

Copyright

Copyright (c) 2024

7.5.2 Function Documentation

7.5.2.1 kiezen()

```
void kiezen (
            command str )
```

Verwerkt een commando en roept de bijbehorende API-functie aan.

Deze functie neemt een commando-string als invoer, bepaalt welke functie moet worden aangeroepen op basis van de hashwaarde van het eerste argument, en roept de juiste API-functie aan met de gegeven argumenten.

Parameters

str	De commando-string die moet worden verwerkt.

Returns

none

7.6 logic_layer.h

Go to the documentation of this file.

```
00001
00014 #ifndef logic_layer_h
00015 #define logic_layer_h
```

```
00016
00017 #include "main.h"
00018 #include "fonts.h"
00019 #include "user_interface.h"
00020
00021 #define lijn 15858359
00022 #define rechthoek 4254663175
00023 #define tekst 532670837
00024 #define bitmap 3993858727
00025 #define clearscherm 96846547
00026 #define wacht 536075777
00027 #define herhaal 1948098847
00028 #define cirkel 4032920196
00029 #define figuur 4149942492
00030 #define polygon 4074764338
00031
00032 void kiezen(command str);
00033
00034 #endif /* logic_layer_h */
```

7.7 VGA_Driver/Core/Inc/main.h File Reference

headerfile of main.c

```
#include "stm32f4xx.h"
#include "vga_driver.h"
#include "stm32_ub_vga_screen.h"
#include "logic_layer.h"
#include "fonts.h"
#include "bitmap.h"
#include "user_interface.h"
#include "uart.h"
#include "stm32f4xx_it.h"
#include <string.h>
#include <stdlib.h>
#include <math.h>
```

Macros

- #define **__STM32F4_UB_MAIN_H**
- #define **BAUD_RATE** 115200

7.7.1 Detailed Description

headerfile of main.c

Author

Michel Vollmuller (michel.vollmuller@gmail.com)

Tim Wannet (tim.wannet@student.hu.nl)

Tijmen Willems (tijmen.willems@student.hu.nl)

Version

0.1

Date

05-06-2024

Copyright

Copyright (c) 2024

7.8 main.h

Go to the documentation of this file.

```
00001 //-------------------------------------------------------------
00002 // File     : main.h
00003 //-------------------------------------------------------------
00004 /* USER CODE BEGIN Header */
00017 /* USER CODE END Header */
00018
00019 /* Define to prevent recursive inclusion -------------------------------------*/
00020 #ifndef __MAIN_H
00021 #define __MAIN_H
00022
00023 #ifdef __cplusplus
00024 extern "C" {
00025 #endif
00026 #endif
00027
00028 /* Private includes ----------------------------------------------------------*/
00029 /* USER CODE BEGIN Includes */
00030
00031 //-------------------------------------------------------------
00032 #ifndef __STM32F4_UB_MAIN_H
00033 #define __STM32F4_UB_MAIN_H
00034
00035 #define BAUD_RATE 115200
00036
00037 //-------------------------------------------------------------
00038 // Includes
00039 //-------------------------------------------------------------
00040 #include "stm32f4xx.h"
00041 #include "vga_driver.h"
00042 #include "stm32_ub_vga_screen.h"
00043 #include "logic_layer.h"
00044 #include "fonts.h"
00045 #include "bitmap.h"
00046 #include "user_interface.h"
00047 #include "stm32_ub_vga_screen.h"
00048 #include "uart.h"
00049 #include "stm32f4xx_it.h"
00050 #include "user_interface.h"
00051
00052 // #include <stdio.h>
00053 #include <string.h>
00054 // #include <stdint.h>
00055 #include <stdlib.h>
00056 #include <math.h>
00057 //-------------------------------------------------------------
00058 #endif // __STM32F4_UB_MAIN_H
```

7.9 VGA_Driver/Core/Inc/stm32_ub_vga_screen.h File Reference

Headerfile of the VGA screen library.

```
#include "stm32f4xx.h"
#include "stm32f4xx_gpio.h"
#include "stm32f4xx_rcc.h"
#include "stm32f4xx_tim.h"
#include "misc.h"
#include "stm32f4xx_dma.h"
```

Data Structures

- struct VGA_t

Macros

- #define **VGA_COL_BLACK** 0x00
- #define **VGA_COL_MAGENTA** 0xE3
- #define **VGA_COL_LIGHT_MAGENTA** 0xEB
- #define **VGA_COL_BLUE** 0x03
- #define **VGA_COL_LIGHT_BLUE** 0x0B
- #define **VGA_COL_CYAN** 0x1A
- #define **VGA_COL_LIGHT_CYAN** 0x1F
- #define **VGA_COL_GREEN** 0x1C
- #define **VGA_COL_LIGHT_GREEN** 0x1D
- #define **VGA_COL_YELLOW** 0xFC
- #define **VGA_COL_RED** 0xE0
- #define **VGA_COL_LIGHT_RED** 0xE4
- #define **VGA_COL_BROWN** 0xAD
- #define **VGA_COL_GREY** 0x05
- #define **VGA_COL_WHITE** 0xFF
- #define **VGA_DISPLAY_X** 320
- #define **VGA_DISPLAY_Y** 240
- #define **VGA_TIM1_PERIODE** 11
- #define **VGA_TIM1_PRESCALE** 0
- #define **VGA_TIM2_HSYNC_PERIODE** 2667
- #define **VGA_TIM2_HSYNC_PRESCALE** 0
- #define **VGA_TIM2_HSYNC_IMP** 320
- #define **VGA_TIM2_HTRIGGER_START** 480
- #define **VGA_TIM2_DMA_DELAY** 60
- #define **VGA_VSYNC_PERIODE** 525
- #define **VGA_VSYNC_IMP** 2
- #define **VGA_VSYNC_BILD_START** 36
- #define **VGA_VSYNC_BILD_STOP** 514
- #define **RAM_SIZE** (VGA_DISPLAY_X+1)∗VGA_DISPLAY_Y
- #define **VGA_GPIOE_BASE_ADR** ((uint32_t)0x40021000)
- #define **VGA_GPIO_ODR_OFFSET** ((uint32_t)0x00000014)
- #define **VGA_GPIO_BYTE_OFFSET** ((uint32_t)0x00000001)
- #define **VGA_GPIOE_ODR_ADDRESS** (VGA_GPIOE_BASE_ADR | VGA_GPIO_ODR_OFFSET | VGA_↩ GPIO_BYTE_OFFSET)
- #define **VGA_GPIO_HINIBBLE** ((uint16_t)0xFF00)

Functions

- void **UB_VGA_Screen_Init** (void)
- void **UB_VGA_FillScreen** (uint8_t color)
- void **UB_VGA_SetPixel** (uint16_t xp, uint16_t yp, uint8_t color)

Variables

- VGA_t **VGA**
- uint8_t **VGA_RAM1** [(VGA_DISPLAY_X+1) ∗VGA_DISPLAY_Y]

7.9.1 Detailed Description

Headerfile of the VGA screen library.

This file contains the main function of the program. It initializes the system, sets up the VGA screen, and enters a loop to handle UART messages.

Authors

Michel Vollmuller, Tim Wannet, Tijmen Willems

Date

5 mei 2024

Version

1.0

7.10 stm32_ub_vga_screen.h

Go to the documentation of this file.

```
00001
00013 //--------------------------------------------------------------
00014 // File     : stm32_ub_vga_screen.h
00015 //--------------------------------------------------------------
00016
00017
00018
00019 //--------------------------------------------------------------
00020 #ifndef __STM32F4_UB_VGA_SCREEN_H
00021 #define __STM32F4_UB_VGA_SCREEN_H
00022
00023
00024 //--------------------------------------------------------------
00025 // Includes
00026 //--------------------------------------------------------------
00027 #include "stm32f4xx.h"
00028 #include "stm32f4xx_gpio.h"
00029 #include "stm32f4xx_rcc.h"
00030 #include "stm32f4xx_tim.h"
00031 #include "misc.h"
00032 #include "stm32f4xx_dma.h"
00033
00034
00035
00036 //--------------------------------------------------------------
00037 // color designation
00038 // 8bit color (R3G3B2)
00039 // Red   (3bit) -> Bit7-Bit5
00040 // Green (3bit) -> Bit4-Bit2
00041 // Blue  (2bit) -> Bit1-Bit0
00042 //--------------------------------------------------------------
00043 #define VGA_COL_BLACK          0x00
00044 #define VGA_COL_MAGENTA        0xE3
00045 #define VGA_COL_LIGHT_MAGENTA  0xEB
00046 #define VGA_COL_BLUE           0x03 // 0x01
00047 #define VGA_COL_LIGHT_BLUE     0x0B
00048 #define VGA_COL_CYAN           0x1A
00049 #define VGA_COL_LIGHT_CYAN     0x1F
00050 #define VGA_COL_GREEN          0x1C
00051 #define VGA_COL_LIGHT_GREEN    0x1D
00052 #define VGA_COL_YELLOW         0xFC
00053 #define VGA_COL_RED            0xE0
00054 #define VGA_COL_LIGHT_RED      0xE4
00055 #define VGA_COL_BROWN          0xAD
00056 #define VGA_COL_GREY           0x05 //0xAF or 0x05
00057 #define VGA_COL_WHITE          0xFF
00058
00059 //--------------------------------------------------------------
```

```
00060 // define the VGA_display
00061 //---------------------------------------------------------------
00062 #define VGA_DISPLAY_X   320
00063 #define VGA_DISPLAY_Y   240
00064
00065 //---------------------------------------------------------------
00066 // VGA Structure
00067 //---------------------------------------------------------------
00068 typedef struct {
00069   uint16_t hsync_cnt;   // counter
00070   uint32_t start_adr;   // start_adres
00071   uint32_t dma2_cr_reg; // Register constant CR-Register
00072 }VGA_t;
00073
00074 extern VGA_t VGA;
00075
00076 extern uint8_t VGA_RAM1[(VGA_DISPLAY_X+1)*VGA_DISPLAY_Y];
00077
00078
00079 //---------------------------------------------------------------
00080 // Timer-1
00081 // Function  = Pixelclock (Speed for  DMA Transfer)
00082 //
00083 // basefreq = 2*APB2 (APB2=84MHz) => TIM_CLK=168MHz
00084 // Frq       = 168MHz/1/12 = 14MHz
00085 //
00086 //---------------------------------------------------------------
00087 #define VGA_TIM1_PERIODE      11
00088 #define VGA_TIM1_PRESCALE      0
00089
00090
00091
00092 //---------------------------------------------------------------
00093 // Timer-2
00094 // Function  = CH4 : HSync-Signal on PB11
00095 //             CH3 : Trigger point for DMA start
00096 //
00097 // basefreq = 2*APB1 (APB1=48MHz) => TIM_CLK=84MHz
00098 // Frq       = 84MHz/1/2668 = 31,48kHz => T = 31,76us
00099 // 1TIC      = 11,90ns
00100 //
00101 //---------------------------------------------------------------
00102 #define  VGA_TIM2_HSYNC_PERIODE   2667
00103 #define  VGA_TIM2_HSYNC_PRESCALE     0
00104
00105 #define  VGA_TIM2_HSYNC_IMP       320  // HSync-length (3,81us)
00106 #define  VGA_TIM2_HTRIGGER_START  480  // HSync+BackPorch (5,71us)
00107 #define  VGA_TIM2_DMA_DELAY        60  // ease the delay when DMA START (Optimization = none)
00108 //#define  VGA_TIM2_DMA_DELAY        30  // ease the delay when DMA START (Optimization = -O1)
00109
00110
00111 //---------------------------------------------------------------
00112 // VSync-Signal
00113 // Trigger   = Timer2 Update (f=31,48kHz => T = 31,76us)
00114 // 1TIC      = 31,76us
00115 //---------------------------------------------------------------
00116 #define  VGA_VSYNC_PERIODE        525
00117 #define  VGA_VSYNC_IMP  2
00118 #define  VGA_VSYNC_BILD_START      36
00119 #define  VGA_VSYNC_BILD_STOP      514   // (16,38ms)
00120 #define RAM_SIZE        (VGA_DISPLAY_X+1)*VGA_DISPLAY_Y
00121
00122
00123 //---------------------------------------------------------------
00124 // Adress from PORTE (Reg ODR) callback DMA
00125 // (see Page 53+204 of the Manual)
00126 //
00127 // Data-Bit0 => PE8
00128 // Data-Bit7 => PE15
00129 //---------------------------------------------------------------
00130 #define VGA_GPIOE_BASE_ADR     ((uint32_t)0x40021000) // ADR from Port-E
00131 #define VGA_GPIO_ODR_OFFSET    ((uint32_t)0x00000014) // ADR from Register ODR
00132 #define VGA_GPIO_BYTE_OFFSET   ((uint32_t)0x00000001) // Data for 8bit
00133 #define VGA_GPIOE_ODR_ADDRESS   (VGA_GPIOE_BASE_ADR | VGA_GPIO_ODR_OFFSET | VGA_GPIO_BYTE_OFFSET)
00134
00135 //---------------------------------------------------------------
00136 // Define for black on PE8 - PE15
00137 //---------------------------------------------------------------
00138 #define VGA_GPIO_HINIBBLE  ((uint16_t)0xFF00) // GPIO_Pin_8 to GPIO_Pin_15
00139
00140 //---------------------------------------------------------------
00141 // Global Function call
00142 //---------------------------------------------------------------
00143 void UB_VGA_Screen_Init(void);
00144 void UB_VGA_FillScreen(uint8_t color);
00145 void UB_VGA_SetPixel(uint16_t xp, uint16_t yp, uint8_t color);
00146
```

```
00147 //-------------------------------------------------------------
00148 #endif // __STM32F4_UB_VGA_SCREEN_H
```

7.11 VGA_Driver/Core/Inc/uart.h File Reference

headerfile of uart.c

```
#include <stdint.h>
#include <stdbool.h>
#include "misc.h"
#include "string.h"
#include "stm32f4xx.h"
#include "stm32f4xx_rcc.h"
```

Macros

- #define **UART_BUFFER_SIZE** 100

Functions

- void UART_Init (uint32_t baudrate)

  Initializes the UART interface.
- void **UART_SendChar** (char c)
- void UART_SendString (char *string)

  Sends a single character over the UART interface. the function blocks untill the character is send.

Variables

- char **UART_RX_message** [UART_BUFFER_SIZE]
- uint16_t **charCnt**
- bool **msgReceivedUSART2**

7.11.1 Detailed Description

headerfile of uart.c

Author

Michel Vollmuller (michel.vollmuller@gmail.com)

Tim Wannet (tim.wannet@student.hu.nl)

Tijmen Willems (tijmen.willems@student.hu.nl)

Version

0.1

Date

05-06-2024

Copyright

Copyright (c) 2024

7.12 uart.h

Go to the documentation of this file.

```
00001
00014 #ifndef UART_H
00015 #define UART_H
00016
00017 #include <stdint.h>
00018 #include <stdbool.h>
00019 #include "misc.h"
00020 #include "string.h"
00021 #include "stm32f4xx.h"
00022 #include "stm32f4xx_rcc.h"
00023 // #include "stm32f4XX_dma.h"
00024
00025 #define UART_BUFFER_SIZE 100
00026
00027 // char UART_TX[UART_BUFFER_SIZE];
00028 extern char UART_RX_message[UART_BUFFER_SIZE];
00029 extern uint16_t charCnt;
00030 extern bool msgReceivedUSART2;
00031
00032 // void UART_Init_DMA(void);
00033 void UART_Init(uint32_t baudrate);
00034 void UART_SendChar (char c);
00035 void UART_SendString (char *string);
00036 // char* UART_GetString (void);
00037
00038 #endif // UART_H
```

7.13 VGA_Driver/Core/Inc/user_interface.h File Reference

headerfile of user_interface.c

Data Structures

- struct command

Typedefs

- typedef struct command **command**

Functions

- command UI_string_to_function (char *str)

  Converts a string to a function name. This function takes a string as input and returns the separated strings The string is tokenized using the comma (',') delimiter.

7.13.1 Detailed Description

headerfile of user_interface.c

Author

Michel Vollmuller (michel.vollmuller@gmail.com)

Tim Wannet (tim.wannet@student.hu.nl)

Tijmen Willems (tijmen.willems@student.hu.nl)

Version

0.1

Date

05-06-2024

Copyright

Copyright (c) 2024

7.13.2 Function Documentation

7.13.2.1 UI_string_to_function()

```
command UI_string_to_function (
            char * str )
```

Converts a string to a function name. This function takes a string as input and returns the separated strings The string is tokenized using the comma (',') delimiter.

Parameters

str	The input string.

Returns

a struct with the strings

7.14 user_interface.h

Go to the documentation of this file.

```
00001
00014 #ifndef user_interface_h
00015 #define user_interface_h
00016
00017
00018 typedef struct command
00019 {
00020     char* arg[20];
00021 } command;
00022
00023 command UI_string_to_function(char* str);
00024
00025
00026
00027 #endif /* user_interface_h */
```

7.15 VGA_Driver/Core/Inc/vga_driver.h File Reference

headerfile of vga_driver.c

Macros

- #define **SMILEY_HAPPY** 1
- #define **SMILEY_SAD** 2
- #define **ARROW_UP** 3
- #define **ARROW_RIGHT** 4
- #define **ARROW_DOWN** 5
- #define **ARROW_LEFT** 6
- #define **MEGAMAN** 7
- #define **MICHIEL** 8
- #define **FRANC** 9
- #define **GROEP** 15
- #define **MAX_LEN_FONTNAME** 30
- #define **LETTERA** 'a'
- #define **LETTERC** 'c'
- #define **arial_hash** 510602739
- #define **Arial_hash** 472653267
- #define **ARIAL_hash** 471467347
- #define **consolas_hash** 1405698636
- #define **Consolas_hash** 3415123500
- #define **CONSOLAS_hash** 3746353484
- #define **NORMAL** 1498505684
- #define **ITALIC** 412296731
- #define **BOLD** 491321
- #define **ARIAL_SMALL_HEIGHT** 10
- #define **ARIAL_SMALL_ITALIC_HEIGHT** 12
- #define **ARIAL_SMALL_BOLD_HEIGHT** 11
- #define **ARIAL_LARGE_HEIGHT** 15
- #define **ARIAL_LARGE_ITALIC_HEIGHT** 15
- #define **ARIAL_LARGE_BOLD_HEIGHT** 16
- #define **SMALL** 1
- #define **LARGE** 2
- #define **ASCII_OFFSET** 32
- #define **BYTE_SIZE** 8
- #define **BITMASK** 128
- #define **ARRAY_DIMENSION** 2
- #define **CHAR_START_OFFSET** 1
- #define **CASE_OFFSET** 32
- #define **ERROR_FONTNAME** 1
- #define **ERROR_FONTNAME_UNKNOWN** 2

Functions

- int API_draw_text (int x_lup, int y_lup, int color, char ∗text, char ∗fontname, int fontsize, int fontstyle, int reserved)

  Draws a string to the VGA screen.
- int API_draw_line (int x1, int y1, int x_2, int y2, int colour, int thickness, int reserved)

  Draw a line on the VGA screen.
- int API_draw_rectangle (int x, int y, int width, int height, int colour, int filled, int reserved1, int reserved2)

  API_draw_rectangle() is used to draw a rectangle to the VGA screen.
- int API_draw_polygon (int x, int y, int size, int corners, int colour, int filled)

API_draw_polygon() is used to draw a polygon to the VGA screen.

- int API_draw_bitmap (int x_lup, int y_lup, int bm_nr)

  Draws a bitmap to the VGA screen.
- int API_clearscreen (int colour)

  API_clearscreen() is used to clear the VGA screen.

- unsigned long hash (char *str)

  hash() is used to hash a string to a unique value.
- uint8_t color_chooser (char *str)

  gives the corresponding colour value for the given string

7.15.1 Detailed Description

headerfile of vga_driver.c

Author

Michel Vollmuller (michel.vollmuller@gmail.com)

Tim Wannet (tim.wannet@student.hu.nl)

Tijmen Willems (tijmen.willems@student.hu.nl)

Version

0.1

Date

05-06-2024

Copyright

Copyright (c) 2024

7.16 vga_driver.h

Go to the documentation of this file.

```
00001
00014 #ifndef vga_driver_h
00015 #define vga_driver_h
00016
00017 #define SMILEY_HAPPY 1
00018 #define SMILEY_SAD 2
00019 #define ARROW_UP 3
00020 #define ARROW_RIGHT 4
00021 #define ARROW_DOWN 5
00022 #define ARROW_LEFT 6
00023 #define MEGAMAN 7
00024 #define MICHIEL 8
00025 #define FRANC 9
00026 #define GROEP 15
00027 #define MAX_LEN_FONTNAME 30
00028 #define LETTERA 'a'
00029 #define LETTERC 'c'
00030
00031 #define arial_hash 510602739
```

```
00032 #define Arial_hash 472653267
00033 #define ARIAL_hash 471467347
00034 #define consolas_hash 1405698636
00035 #define Consolas_hash 3415123500
00036 #define CONSOLAS_hash 3746353484
00037
00038 #define NORMAL 1498505684
00039 #define ITALIC 412296731
00040 #define BOLD 491321
00041
00042 #define ARIAL_SMALL_HEIGHT        10
00043 #define ARIAL_SMALL_ITALIC_HEIGHT 12
00044 #define ARIAL_SMALL_BOLD_HEIGHT   11
00045 #define ARIAL_LARGE_HEIGHT          15
00046 #define ARIAL_LARGE_ITALIC_HEIGHT   15
00047 #define ARIAL_LARGE_BOLD_HEIGHT     16
00048
00049 #define SMALL 1
00050 #define LARGE 2
00051 #define ASCII_OFFSET 32
00052 #define BYTE_SIZE 8
00053 #define BITMASK 128
00054 #define ARRAY_DIMENSION 2
00055 #define CHAR_START_OFFSET 1
00056 #define CASE_OFFSET      32
00057 #define ERROR_FONTNAME   1
00058 #define ERROR_FONTNAME_UNKNOWN 2
00059
00060 extern int API_draw_text (int x_lup, int y_lup, int color, char *text, char *fontname,int fontsize,
      int fontstyle, int reserved); // fontsize:1 small, 2 big
00061 extern int API_draw_line (int x1, int y1, int x_2, int y2, int colour, int thickness, int reserved);
00062 extern int API_draw_rectangle (int x, int y, int width, int height, int colour, int filled, int
      reserved1, int reserved2); // e.g.: weight, bordercolor
00063 extern int API_draw_polygon (int x, int y, int size, int corners, int colour, int filled);
00064 extern int API_draw_bitmap (int x_lup, int y_lup, int bm_nr);
00065 extern int API_clearscreen (int colour);
00066 extern unsigned long hash(char *str);
00067 extern uint8_t color_chooser(char *str);
00068 #endif /* vga_driver_h */
```

7.17 VGA_Driver/Core/Src/bitmap.c File Reference

This file contains different bitmap images.

```
#include "main.h"
```

Variables

- const unsigned char megaman [ ]

  Bitmap data for the "Megaman" sprite.
- const uint8_t megaman_2 [504]

  A bitmap representation of an megaman image.
- const uint8_t smiley_sad [SMILEY_WIDTH ∗SMILEY_HEIGHT]

  A bitmap representation of a sad smiley image.
- const uint8_t smiley_happy [SMILEY_WIDTH ∗SMILEY_HEIGHT]

  A bitmap representation of a happy smiley image.
- const uint8_t arrow_up [27 ∗41]

  A bitmap representation of a upward arrow image.
- const uint8_t arrow_down [ARROW_DOWN_WIDTH ∗ARROW_DOWN_HEIGHT]

  A bitmap representation of a downward arrow image.
- const uint8_t arrow_left [ARROW_LEFT_WIDTH ∗ARROW_LEFT_HEIGHT]

  A bitmap representation of a left arrow image.
- const uint8_t arrow_right [ARROW_RIGHT_WIDTH ∗ARROW_RIGHT_HEIGHT]

  A bitmap representation of a right arrow image.

- const uint8_t michiel [MICHIEL_WIDTH ∗MICHIEL_HEIGHT]

  A bitmap representation of Michiel Scager.
- const uint8_t franc [FRANC_WIDTH ∗FRANC_HEIGHT]

  A bitmap representation of Franc.
- const uint8_t groep [GROEP_WIDTH ∗GROEP_HEIGHT]

  A bitmap representation of our group image.

7.17.1 Detailed Description

This file contains different bitmap images.

Author

Michel Vollmuller, Tim Wannet, Tijmen Willems

Date

22-05-2024

Precondition

This file must be used in combination with main.h

7.17.2 Variable Documentation

7.17.2.1 arrow_down

```
const uint8_t arrow_down[ARROW_DOWN_WIDTH *ARROW_DOWN_HEIGHT]
```

A bitmap representation of a downward arrow image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants ARROW_DOWN_WIDTH and ARROW_DOWN_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.17.2.2 arrow_left

```
const uint8_t arrow_left[ARROW_LEFT_WIDTH *ARROW_LEFT_HEIGHT]
```

Initial value:

```
= {
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xff,
      0xfe, 0xff, 0xff, 0xfe, 0xff, 0xff, 0x71, 0xff, 0xfe, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xfe,
      0xfc, 0xfb, 0xff, 0xff, 0xa8, 0x00, 0x00, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xff,
      0xfe, 0xff, 0xe0, 0x18, 0x00, 0x10, 0x05, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xfd, 0xff,
      0xff, 0x48, 0x00, 0x0f, 0x11, 0x10, 0x07, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
```

```
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xff, 0xfe, 0xff, 0xff, 0x7c,
  0x00, 0x08, 0x11, 0x11, 0x12, 0x10, 0x07, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xff, 0xfa, 0xff, 0xff, 0xb9, 0x00, 0x00,
  0x12, 0x11, 0x12, 0x12, 0x11, 0x10, 0x07, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xe7, 0x26, 0x00, 0x10, 0x11,
  0x11, 0x11, 0x12, 0x12, 0x11, 0x10, 0x07, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xfe, 0xfd, 0xfc, 0xfd, 0xff, 0xff, 0x51, 0x00, 0x0d, 0x11, 0x11, 0x11,
  0x12, 0x12, 0x12, 0x12, 0x11, 0x10, 0x07, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x8a, 0x00, 0x05, 0x11, 0x11, 0x11, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x11, 0x10, 0x07, 0xff, 0xfe, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd,
  0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfe, 0xfe,
    0xff, 0xfe, 0xfe, 0xfd, 0xfc, 0xff, 0xc4, 0x0b, 0x00, 0x10, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x12, 0x10, 0x07, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xfe, 0xfe, 0xff, 0xfb, 0x3e, 0x00, 0x10, 0x10, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x12, 0x11, 0x10, 0x11, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e,
  0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0b, 0x37,
    0xfa, 0xff, 0xff, 0x67, 0x0c, 0x04, 0x13, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x11, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10,
  0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x0e, 0x34,
    0xff, 0x92, 0x00, 0x20, 0x0c, 0x13, 0x0f, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11,
  0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x0f, 0x35,
    0x5b, 0x00, 0x2e, 0x09, 0x12, 0x10, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x12, 0x11, 0x10, 0x35,
    0xff, 0xcf, 0x02, 0x00, 0x11, 0x11, 0x12, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x11, 0x11, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10,
  0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x0e, 0x35,
    0xfc, 0xff, 0xff, 0x8e, 0x00, 0x07, 0x11, 0x13, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x12, 0x11, 0x10, 0x0f, 0x0f, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e,
  0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0c, 0x32,
    0xfe, 0xfe, 0xfc, 0xff, 0xff, 0x59, 0x00, 0x0b, 0x11, 0x11, 0x12, 0x11, 0x12, 0x12, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x12, 0x10, 0x0c, 0x30, 0x2c, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d,
  0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2a, 0x53,
    0xff, 0xff, 0xfe, 0xfe, 0xff, 0xff, 0xe4, 0x26, 0x00, 0x11, 0x10, 0x13, 0x11, 0x12, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x11, 0x11, 0x08, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xfe, 0xfd, 0xfe, 0xff, 0xb6, 0x00, 0x00, 0x11, 0x12, 0x11, 0x12, 0x12,
  0x12, 0x12, 0x12, 0x12, 0x11, 0x10, 0x07, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xfd, 0xff, 0xff, 0x83, 0x00, 0x08, 0x11, 0x11, 0x11,
  0x12, 0x12, 0x12, 0x12, 0x11, 0x10, 0x07, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xfe, 0x4e, 0x00, 0x0d, 0x11,
  0x11, 0x12, 0x12, 0x12, 0x11, 0x10, 0x07, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xdd, 0x1b, 0x00,
  0x12, 0x12, 0x12, 0x11, 0x12, 0x10, 0x07, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xfe, 0xfb, 0xff, 0xff, 0xa8,
  0x00, 0x00, 0x12, 0x11, 0x12, 0x10, 0x06, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xfe, 0xfe, 0xff,
  0xff, 0x78, 0x00, 0x0c, 0x12, 0x11, 0x06, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe, 0xfe,
  0xfc, 0xff, 0xf9, 0x3f, 0x00, 0x0d, 0x06, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xfd, 0xfe, 0xff, 0xff, 0xce, 0x12, 0x00, 0xff, 0xfe, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xfe, 0xff, 0xff, 0x9b, 0xff, 0xfe, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
  0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
}
```

A bitmap representation of a left arrow image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants ARROW_LEFT_WIDTH and ARROW_LEFT_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.17.2.3 arrow_right

```
const uint8_t arrow_right[ARROW_RIGHT_WIDTH *ARROW_RIGHT_HEIGHT]
```

Initial value:

```
= {
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xfe, 0xff, 0x9b, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xfe, 0xff, 0x00, 0x12, 0xce, 0xff, 0xff, 0xfe, 0xfd, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x06, 0x0d, 0x00, 0x3f, 0xf9, 0xff, 0xfc, 0xfe, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x06, 0x11, 0x12, 0x0c, 0x00, 0x78, 0xff, 0xff, 0xfe, 0xfe, 0xfe, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x06, 0x10, 0x12, 0x11, 0x12, 0x00, 0x00, 0xa8, 0xff, 0xff, 0xfb, 0xfe, 0xfe, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x07, 0x10, 0x12, 0x11, 0x12, 0x12, 0x12, 0x00, 0x1b, 0xdd, 0xff, 0xff, 0xfe, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x07, 0x10, 0x11, 0x12, 0x12, 0x12, 0x11, 0x11, 0x0d, 0x00, 0x4e, 0xfe, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x07, 0x10, 0x11, 0x12, 0x12, 0x12, 0x12, 0x11, 0x11, 0x11, 0x08, 0x00, 0x83, 0xff, 0xff,
    0xfd, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x07, 0x10, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x11, 0x12, 0x11, 0x00, 0x00, 0xb6,
    0xff, 0xfe, 0xfd, 0xfe, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0x08, 0x11, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x11, 0x13, 0x10, 0x11, 0x00,
    0x26, 0xe4, 0xff, 0xff, 0xfe, 0xfe, 0xff, 0xff,
      0x53, 0x2a, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d, 0x2d,
    0x2c, 0x30, 0x0c, 0x10, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x11, 0x12, 0x11, 0x11,
    0x0b, 0x00, 0x59, 0xff, 0xff, 0xfc, 0xfe, 0xfe,
      0x32, 0x0c, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e,
    0x0f, 0x0f, 0x10, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x11,
    0x13, 0x11, 0x07, 0x00, 0x8e, 0xff, 0xff, 0xfc,
      0x35, 0x0e, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10,
    0x11, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
    0x11, 0x12, 0x11, 0x11, 0x00, 0x02, 0xcf, 0xff,
      0x35, 0x10, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
    0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
    0x12, 0x12, 0x10, 0x12, 0x09, 0x2e, 0x00, 0x5b,
      0x35, 0x0f, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11, 0x11,
    0x11, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
    0x11, 0x0f, 0x13, 0x0c, 0x20, 0x00, 0x92, 0xff,
      0x34, 0x0e, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10, 0x10,
    0x10, 0x10, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12,
    0x11, 0x13, 0x04, 0x0c, 0x67, 0xff, 0xff, 0xfa,
      0x37, 0x0b, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e, 0x0e,
    0x0e, 0x11, 0x10, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x11, 0x10,
    0x10, 0x00, 0x3e, 0xfb, 0xff, 0xfe, 0xfe, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0x07, 0x10, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x10, 0x00,
    0x0b, 0xc4, 0xff, 0xfc, 0xfd, 0xfe, 0xfe, 0xff,
      0xfe, 0xfe, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd, 0xfd,
    0xfe, 0xff, 0x07, 0x10, 0x11, 0x12, 0x12, 0x12, 0x12, 0x12, 0x12, 0x11, 0x11, 0x11, 0x05, 0x00, 0x8a,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x07, 0x10, 0x11, 0x12, 0x12, 0x12, 0x12, 0x11, 0x11, 0x11, 0x0d, 0x00, 0x51, 0xff, 0xff,
    0xfd, 0xfc, 0xfd, 0xfe, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x07, 0x10, 0x11, 0x12, 0x12, 0x11, 0x11, 0x11, 0x10, 0x00, 0x26, 0xe7, 0xff, 0xff, 0xff,
    0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x07, 0x10, 0x11, 0x12, 0x12, 0x11, 0x12, 0x00, 0x00, 0xb9, 0xff, 0xff, 0xfa, 0xff, 0xfe,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x07, 0x10, 0x12, 0x11, 0x11, 0x08, 0x00, 0x7c, 0xff, 0xff, 0xfe, 0xff, 0xfe, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x07, 0x10, 0x11, 0x0f, 0x00, 0x48, 0xff, 0xff, 0xfd, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x05, 0x10, 0x00, 0x18, 0xe0, 0xff, 0xfe, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xff, 0xff, 0x00, 0x00, 0xa8, 0xff, 0xff, 0xfb, 0xfc, 0xfe, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
      0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xfe,
    0xfe, 0xff, 0x71, 0xff, 0xff, 0xfe, 0xff, 0xff, 0xfe, 0xff, 0xfe, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff
}
```

A bitmap representation of a right arrow image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants ARROW_RIGHT_WIDTH and ARROW_RIGHT_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.17.2.4 arrow_up

```
const uint8_t arrow_up[27 *41]
```

A bitmap representation of a upward arrow image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants ARROW_UP_WIDTH and ARROW_UP_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.17.2.5 franc

```
const uint8_t franc[FRANC_WIDTH *FRANC_HEIGHT]
```

A bitmap representation of Franc.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants FRANC_WIDTH and FRANC_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.17.2.6 groep

```
const uint8_t groep[GROEP_WIDTH *GROEP_HEIGHT]
```

A bitmap representation of our group image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants GROUP_WIDTH and GROUP_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.17.2.7 megaman

```
const unsigned char megaman[ ]
```

Initial value:

```
= {
    0B11111111, 0B11000111, 0B11111111,
    0B11111111, 0B00000011, 0B11111111,
    0B11111110, 0B00001001, 0B11111111,
    0B11111100, 0B00000000, 0B11111111,
    0B11111100, 0B00000000, 0B01111111,
    0B11111000, 0B00000000, 0B01111111,
    0B11111000, 0B01111001, 0B01111111,
    0B11111000, 0B11100101, 0B01111111,
    0B11111100, 0B11100101, 0B01111111,
    0B11111000, 0B11111111, 0B01111111,
    0B11100010, 0B01000010, 0B00111111,
    0B11001111, 0B01111101, 0B10011111,
    0B11000101, 0B10000001, 0B00011111,
    0B10000000, 0B11111000, 0B00001111,
    0B10000000, 0B11111100, 0B00001111,
    0B10000000, 0B11111000, 0B00001111,
    0B10000000, 0B00000000, 0B00001111,
    0B11000100, 0B00000001, 0B00011111,
```

```
    0B11111001, 0B00001100, 0B11111111,
    0B11110001, 0B10001110, 0B01111111,
    0B11100000, 0B00100000, 0B00111111,
    0B10000000, 0B01110000, 0B00001111,
    0B00000000, 0B01110000, 0B00000111,
    0B00000000, 0B01110000, 0B00000111
}
```

Bitmap data for the "Megaman" sprite.

This array contains the bitmap data for the "Megaman" sprite. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of the sprite. The height and width of the sprite are determined by the size of this array and the specific layout of the bitmap data.

7.17.2.8 megaman_2

```
const uint8_t megaman_2[504]
```

Initial value:

```
= {
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x8c, 0x8d, 0x00, 0xff, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x6e, 0x6e, 0x6e, 0x01, 0x8d, 0x8e, 0x00, 0xff,
    0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x6f, 0x6e, 0x6e, 0x6e, 0x6e, 0x00, 0x00, 0x00, 0x00,
    0xff, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x6f, 0x6e, 0x6e, 0x6e, 0x6e, 0x00, 0x8d, 0x8e, 0x6f,
    0x00, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x8f, 0x6e, 0x6e, 0x6d, 0x6d, 0x6d, 0x6c, 0x00, 0x00, 0x6f,
    0x00, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x8f, 0x6e, 0x6d, 0xe3, 0xfe, 0xff, 0xfe, 0x6c, 0x69, 0xff,
    0x00, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x90, 0x6d, 0xe3, 0xfe, 0xff, 0x00, 0x00, 0xe4, 0x00, 0xff,
    0x00, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x6f, 0xe3, 0xfe, 0xff, 0x00, 0x00, 0xe4, 0x00, 0xff,
    0x00, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xff, 0x00, 0x00, 0x70, 0xe5, 0xfe, 0xff, 0xff, 0xfe, 0xe4, 0xe3, 0xe6,
    0x00, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0x00, 0x00, 0x90, 0x8d, 0x00, 0x6f, 0xe3, 0x00, 0x00, 0x00, 0x00, 0xe6, 0x00,
    0x00, 0x00, 0xff, 0xff, 0xff,
        0xff, 0xff, 0x00, 0x6b, 0x8e, 0x8d, 0x8d, 0x8c, 0x00, 0xe5, 0xe4, 0xe4, 0xe4, 0xe6, 0x00, 0x8c,
    0x8e, 0x6e, 0x00, 0xff, 0xff,
        0xff, 0xff, 0x00, 0x6b, 0x6e, 0x8e, 0x8f, 0x8d, 0x8c, 0x00, 0x00, 0x00, 0x00, 0x00, 0x8f, 0x8d,
    0x6e, 0x6d, 0x00, 0xff, 0xff,
        0xff, 0x00, 0x6a, 0x6e, 0x6f, 0x8f, 0x00, 0x8f, 0x8d, 0x8d, 0x8d, 0x8d, 0x8d, 0x8e, 0x00, 0x90,
    0x6f, 0x6e, 0x6c, 0x00, 0xff,
        0xff, 0x00, 0x6a, 0x6e, 0x00, 0x00, 0x00, 0x8f, 0x8d, 0x8d, 0x8d, 0x8d, 0x8d, 0x8e, 0x00, 0x00,
    0x00, 0x6e, 0x6c, 0x00, 0xff,
        0xff, 0x00, 0x6a, 0x6e, 0x6e, 0x00, 0x00, 0x8e, 0x8d, 0x8d, 0x8d, 0x8d, 0x8d, 0x8f, 0x00, 0x00,
    0x6e, 0x6e, 0x6c, 0x00, 0xff,
        0xff, 0x00, 0x6a, 0x6e, 0x6e, 0x00, 0x00, 0x6f, 0x6e, 0x6e, 0x6e, 0x6e, 0x6e, 0x6f, 0x00, 0x00,
    0x6e, 0x6e, 0x6c, 0x00, 0xff,
        0xff, 0xff, 0x00, 0x00, 0x00, 0xfd, 0x04, 0x70, 0x6e, 0x6e, 0x6e, 0x6e, 0x6f, 0x70, 0x00, 0xfd,
    0x00, 0x00, 0x00, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xff, 0xf9, 0x00, 0x8e, 0x8d, 0x70, 0x70, 0x6c, 0x70, 0x8d, 0x8d, 0x90, 0x00,
    0xf9, 0xff, 0xff, 0xff, 0xff,
        0xff, 0xff, 0xff, 0xf7, 0x00, 0x6f, 0x6e, 0x8c, 0x8c, 0x89, 0x00, 0x8a, 0x8d, 0x8b, 0x8c, 0x6f,
    0x00, 0xf7, 0xff, 0xff, 0xff,
        0xff, 0xf6, 0xf5, 0x00, 0x00, 0x6f, 0x6e, 0x6e, 0x88, 0x00, 0xff, 0x00, 0x88, 0x6e, 0x6e, 0x6f,
    0x00, 0x00, 0xf5, 0xf6, 0xff,
        0xf4, 0x00, 0x00, 0x6e, 0x6e, 0x6e, 0x6e, 0x6e, 0x00, 0xff, 0xff, 0xff, 0x00, 0x6e, 0x6e, 0x6e,
    0x6e, 0x6e, 0x00, 0x00, 0xf4,
        0x00, 0x63, 0x67, 0x67, 0x67, 0x67, 0x67, 0x67, 0x00, 0xff, 0xff, 0xff, 0x00, 0x67, 0x67, 0x67,
    0x67, 0x67, 0x67, 0x63, 0x00,
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xff, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x00, 0x00, 0x00
}
```

A bitmap representation of an megaman image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image is likely to be 21x24 pixels (504 elements) given the size of the array. The color of each pixel is represented as an 8-bit unsigned integer.

7.17.2.9 michiel

```
const uint8_t michiel[MICHIEL_WIDTH *MICHIEL_HEIGHT]
```

A bitmap representation of Michiel Scager.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants MICHIEL_WIDTH and MICHIEL_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.17.2.10 smiley_happy

```
const uint8_t smiley_happy[SMILEY_WIDTH *SMILEY_HEIGHT]
```

A bitmap representation of a happy smiley image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants SMILEY_WIDTH and SMILEY_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.17.2.11 smiley_sad

```
const uint8_t smiley_sad[SMILEY_WIDTH *SMILEY_HEIGHT]
```

A bitmap representation of a sad smiley image.

This array represents a bitmap image where each element corresponds to a pixel's color in the image. The image dimensions are defined by the constants SMILEY_WIDTH and SMILEY_HEIGHT. The color of each pixel is represented as an 8-bit unsigned integer.

7.18 VGA_Driver/Core/Src/fonts.c File Reference

For al the bitmaps of the different fonts.

```
#include "main.h"
```

Variables

- const uint8_t arial_8ptBitmaps [ ]

  Bitmap data for Arial 8pt font.
- const uint16_t arial_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 8pt font.
- const uint8_t arial_italic_8ptBitmaps [ ]

  Bitmap data for Arial 8pt italic font.
- const uint16_t arial_italic_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 8pt italic font.
- const uint8_t arial_bold_8ptBitmaps [ ]

  Bitmap data for Arial 8pt bold font.
- const uint16_t arial_bold_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 8pt bold font.
- const uint8_t arial_11ptBitmaps [ ]

  Bitmap data for Arial 11pt font.
- const uint16_t arial_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 11pt font.
- const uint8_t arial_italic_11ptBitmaps [ ]

  Bitmap data for Arial 11pt italic font.
- const uint16_t arial_italic_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 11pt italic font.
- const uint8_t arial_bold_11ptBitmaps [ ]

  Bitmap data for Arial 11pt bold font.
- const uint16_t arial_bold_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Arial 11pt bold font.
- const uint8_t consolas_8ptBitmaps [ ]

  Bitmap data for Consolas 8pt font.
- const uint16_t consolas_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 8pt font.
- const uint8_t consolas_italic_8ptBitmaps [ ]

  Bitmap data for Consolas 8pt italic font.
- const uint16_t consolas_italic_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 8pt italic font.
- const uint8_t consolas_bold_8ptBitmaps [ ]

  Bitmap data for Consolas 8pt bold font.
- const uint16_t consolas_bold_8ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 8pt bold font.
- const uint8_t consolas_11ptBitmaps [ ]

  Bitmap data for Consolas 11pt font.
- const uint16_t consolas_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 11pt font.
- const uint8_t consolas_italic_11ptBitmaps [ ]

  Bitmap data for Consolas 11pt italic font.
- const uint16_t consolas_italic_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 11pt italic font.
- const uint8_t consolas_bold_11ptBitmaps [ ]

  Bitmap data for Consolas 11pt bold font.
- const uint16_t consolas_bold_11ptDescriptors [NR_OF_SYMBOLS][NR_OF_ELEMENTS]

  Descriptors for Consolas 11pt bold font.

7.18.1 Detailed Description

For al the bitmaps of the different fonts.

This file contains the main function of the program. It initializes the system, sets up the VGA screen, and enters a loop to handle UART messages.

Authors

Michel Vollmuller, Tim Wannet, Tijmen Willems

Date

5 mei 2024

Version

1.0

Precondition

This file must be used in combination with main.h

7.18.2 Variable Documentation

7.18.2.1 arial_11ptBitmaps

```
const uint8_t arial_11ptBitmaps[]
```

Bitmap data for Arial 11pt font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.2 arial_11ptDescriptors

```
const uint16_t arial_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Arial 11pt font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.18.2.3 arial_8ptBitmaps

```
const uint8_t arial_8ptBitmaps[]
```

Bitmap data for Arial 8pt font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.4 arial_8ptDescriptors

```
const uint16_t arial_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Arial 8pt font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with NR_OF_SYMBOLS rows and NR_OF_ELEMENTS columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.18.2.5 arial_bold_11ptBitmaps

```
const uint8_t arial_bold_11ptBitmaps[]
```

Bitmap data for Arial 11pt bold font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.6 arial_bold_11ptDescriptors

```
const uint16_t arial_bold_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Arial 11pt bold font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with NR_OF_SYMBOLS rows and NR_OF_ELEMENTS columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.18.2.7 arial_bold_8ptBitmaps

```
const uint8_t arial_bold_8ptBitmaps[]
```

Bitmap data for Arial 8pt bold font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.8 arial_bold_8ptDescriptors

```
const uint16_t arial_bold_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Arial 8pt bold font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with NR_OF_SYMBOLS rows and NR_OF_ELEMENTS columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.18.2.9 arial_italic_11ptBitmaps

```
const uint8_t arial_italic_11ptBitmaps[]
```

Bitmap data for Arial 11pt italic font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.10 arial_italic_11ptDescriptors

```
const uint16_t arial_italic_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Arial 11pt italic font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with NR_OF_SYMBOLS rows and NR_OF_ELEMENTS columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.18.2.11 arial_italic_8ptBitmaps

```
const uint8_t arial_italic_8ptBitmaps[]
```

Bitmap data for Arial 8pt italic font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.12 arial_italic_8ptDescriptors

```
const uint16_t arial_italic_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Arial 8pt italic font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.18.2.13 consolas_11ptBitmaps

```
const uint8_t consolas_11ptBitmaps[]
```

Bitmap data for Consolas 11pt font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.14 consolas_11ptDescriptors

```
const uint16_t consolas_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Consolas 11pt font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.18.2.15 consolas_8ptBitmaps

```
const uint8_t consolas_8ptBitmaps[]
```

Bitmap data for Consolas 8pt font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.16 consolas_8ptDescriptors

```
const uint16_t consolas_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Consolas 8pt font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with NR_OF_SYMBOLS rows and NR_OF_ELEMENTS columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.18.2.17 consolas_bold_11ptBitmaps

```
const uint8_t consolas_bold_11ptBitmaps[]
```

Bitmap data for Consolas 11pt bold font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.18 consolas_bold_11ptDescriptors

```
const uint16_t consolas_bold_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Consolas 11pt bold font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with NR_OF_SYMBOLS rows and NR_OF_ELEMENTS columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.18.2.19 consolas_bold_8ptBitmaps

```
const uint8_t consolas_bold_8ptBitmaps[]
```

Bitmap data for Consolas 8pt bold font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.20 consolas_bold_8ptDescriptors

```
const uint16_t consolas_bold_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Consolas 8pt bold font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.18.2.21 consolas_italic_11ptBitmaps

```
const uint8_t consolas_italic_11ptBitmaps[]
```

Bitmap data for Consolas 11pt italic font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.22 consolas_italic_11ptDescriptors

```
const uint16_t consolas_italic_11ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Consolas 11pt italic font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.18.2.23 consolas_italic_8ptBitmaps

```
const uint8_t consolas_italic_8ptBitmaps[]
```

Bitmap data for Consolas 8pt italic font.

This array contains the bitmap data for the Arial 8pt font. Each byte in the array represents 8 pixels, with the MSB representing the leftmost pixel and the LSB representing the rightmost pixel.

The data is organized as a series of rows, with each row representing a line of text. The height of each row is determined by the font size (8pt in this case).

The width of each row (i.e., the number of bytes per row) is determined by the number of characters in the font and the width of each character.

7.18.2.24 consolas_italic_8ptDescriptors

```
const uint16_t consolas_italic_8ptDescriptors[NR_OF_SYMBOLS][NR_OF_ELEMENTS]
```

Descriptors for Consolas 8pt italic font.

This array contains the descriptors for the Arial 8pt font. Each descriptor provides information about a specific character in the font, such as its width and the offset of its bitmap data in the arial_8ptBitmaps array.

The array is organized as a 2D array with `NR_OF_SYMBOLS` rows and `NR_OF_ELEMENTS` columns. Each row corresponds to a character in the font, and the columns provide the following information for each character:

0: The width of the character in pixels. 1: The vertical offset of the character's bitmap data in the arial_8ptBitmaps array.

7.19 VGA_Driver/Core/Src/logic_layer.c File Reference

Logic Layer.

```
#include "main.h"
```

Functions

- void kiezen (command str)

  Verwerkt een commando en roept de bijbehorende API-functie aan.

7.19.1 Detailed Description

Logic Layer.

Authors

Michel Vollmuller, Tim Wannet, Tijmen Willems

Date

8 mei 2024

Version

1.0

Precondition

This file must be used in combination with main.h

7.19.2 Function Documentation

7.19.2.1 kiezen()

```
void kiezen (
            command str )
```

Verwerkt een commando en roept de bijbehorende API-functie aan.

Deze functie neemt een commando-string als invoer, bepaalt welke functie moet worden aangeroepen op basis van de hashwaarde van het eerste argument, en roept de juiste API-functie aan met de gegeven argumenten.

Parameters

str	De commando-string die moet worden verwerkt.

Returns

none

7.20 VGA_Driver/Core/Src/main.c File Reference

main.c

```
#include "main.h"
```

Functions

- int main (void)

  Main function of the program.

7.20.1 Detailed Description

main.c

This file contains the main function of the program. It initializes the system, sets up the VGA screen, and enters a loop to handle UART messages.

Authors

Michel Vollmuller, Tim Wannet, Tijmen Willems

Date

5 mei 2024

Version

1.0

Precondition

This file must be used in combination with main.h

7.20.2 Function Documentation

7.20.2.1 main()

```
int main (
            void  )
```

Main function of the program.

This function initializes the system, sets up the VGA screen, and enters a loop to handle UART messages.

Returns

no return value

7.21 VGA_Driver/Core/Src/stm32_ub_vga_screen.c File Reference

stm32_ub_vga_screen.c

```
#include "stm32_ub_vga_screen.h"
```

Functions

- void **P_VGA_InitIO** (void)
- void **P_VGA_InitTIM** (void)
- void **P_VGA_InitINT** (void)
- void **P_VGA_InitDMA** (void)
- void **UB_VGA_Screen_Init** (void)
- void **UB_VGA_FillScreen** (uint8_t color)
- void **UB_VGA_SetPixel** (uint16_t xp, uint16_t yp, uint8_t color)
- void **TIM2_IRQHandler** (void)
- void **DMA2_Stream5_IRQHandler** (void)

Variables

- VGA_t **VGA**
- uint8_t **VGA_RAM1** [(VGA_DISPLAY_X+1) ∗VGA_DISPLAY_Y]

7.21.1 Detailed Description

stm32_ub_vga_screen.c

Authors

Michel Vollmuller, Tim Wannet, Tijmen Willems

Date

5 mei 2024

Version

1.0

7.22 VGA_Driver/Core/Src/uart.c File Reference

This file contains the implementation of the uart communication.

```
#include "uart.h"
```

Functions

- void **USART2_IRQHandler** (void)
- void UART_Init (uint32_t baudrate)

  Initializes the UART interface.
- void **UART_SendChar** (char c)
- void UART_SendString (char ∗string)

  Sends a single character over the UART interface. the function blocks untill the character is send.

Variables

- char **UART_TX_message** [UART_BUFFER_SIZE]
- char **UART_RX_message** [UART_BUFFER_SIZE]
- uint16_t **charCnt** = 0
- bool **msgReceivedUSART2** = false

7.22.1 Detailed Description

This file contains the implementation of the uart communication.

Authors

Michel Vollmuller, Tim Wannet, Tijmen Willems

Date

5 mei 2024

Version

1.0

Precondition

This file must be used in combination with main.h

7.23 VGA_Driver/Core/Src/user_interface.c File Reference

This file contains the implementation of the user interface functions.

```
#include "main.h"
```

Functions

- command UI_string_to_function (char *str)

  Converts a string to a function name. This function takes a string as input and returns the separated strings The string is tokenized using the comma (',') delimiter.

7.23.1 Detailed Description

This file contains the implementation of the user interface functions.

Authors

Michel Vollmuller, Tim Wannet, Tijmen Willems

Date

5 mei 2024

Version

1.0

Precondition

This file must be used in combination with main.h

7.23.2 Function Documentation

7.23.2.1 UI_string_to_function()

```
command UI_string_to_function (
            char * str )
```

Converts a string to a function name. This function takes a string as input and returns the separated strings The string is tokenized using the comma (',') delimiter.

Parameters

str	The input string.

Returns

a struct with the strings

7.24 VGA_Driver/Core/Src/vga_driver.c File Reference

This file contains the implementation of vga driver functions.

```
#include "main.h"
```

Macros

- #define **ZWART** 540422306
- #define **LICHTMAGENTA** 338820699
- #define **MAGENTA** 3940791655
- #define **BLAUW** 511564997
- #define **LICHTBLAUW** 1778211001
- #define **CYAAN** 513217430
- #define **LICHTCYAAN** 1779863434
- #define **GROEN** 517724933
- #define **LICHTGROEN** 1784370937
- #define **GEEL** 15674151
- #define **ROOD** 16080670
- #define **LICHTROOD** 4089130130
- #define **BRUIN** 511801994
- #define **GRIJS** 517718569
- #define **WIT** 492542
- #define **PI** 3.14159265
- #define **TERMS** 4
- #define **UNUSED**(x) (void)(x)

Functions

- int API_draw_text (int x_lup, int y_lup, int color, char *text, char *fontname, int fontsize, int fontstyle, int reserved)

  Draws a string to the VGA screen.
- int API_draw_line (int x1, int y1, int x2, int y2, int colour, int thickness, int reserved)

  Draw a line on the VGA screen.
- int API_draw_rectangle (int x, int y, int width, int height, int colour, int filled, int reserved1, int reserved2)

  API_draw_rectangle() is used to draw a rectangle to the VGA screen.
- int API_draw_polygon (int x, int y, int size, int corners, int colour, int reserved)

  API_draw_polygon() is used to draw a polygon to the VGA screen.
- int API_draw_bitmap (int x_lup, int y_lup, int bm_nr)

  Draws a bitmap to the VGA screen.
- int API_clearscreen (int colour)

  API_clearscreen() is used to clear the VGA screen.
- unsigned long hash (char *str)

  hash() is used to hash a string to a unique value.
- uint8_t color_chooser (char *str)

  gives the corresponding colour value for the given string

Variables

- const double cos_table [73]
- const double sin_table [73]

7.24.1 Detailed Description

This file contains the implementation of vga driver functions.

Author

Michel Vollmuller, Tim Wannet, Tijmen Willems

Date

5 mei 2024

Version

1.0

Precondition

This file must be used in combination with main.h

Index