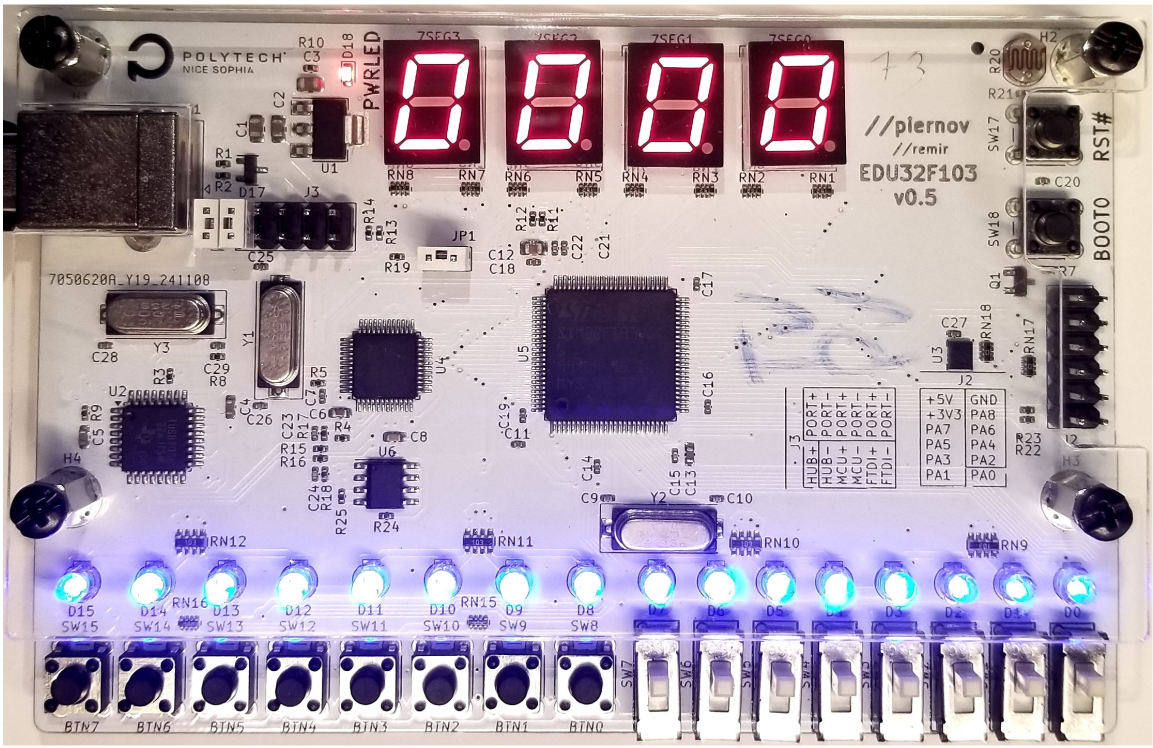
EDU32F103 v0.5
User Manual v0.2

Pierre-Emmanuel Novac <penovac@unice.fr>

25 November 2024



https://leat-edge.github.io/EDU32F103/

Contents

List of Figures

List of Tables

1 Introduction

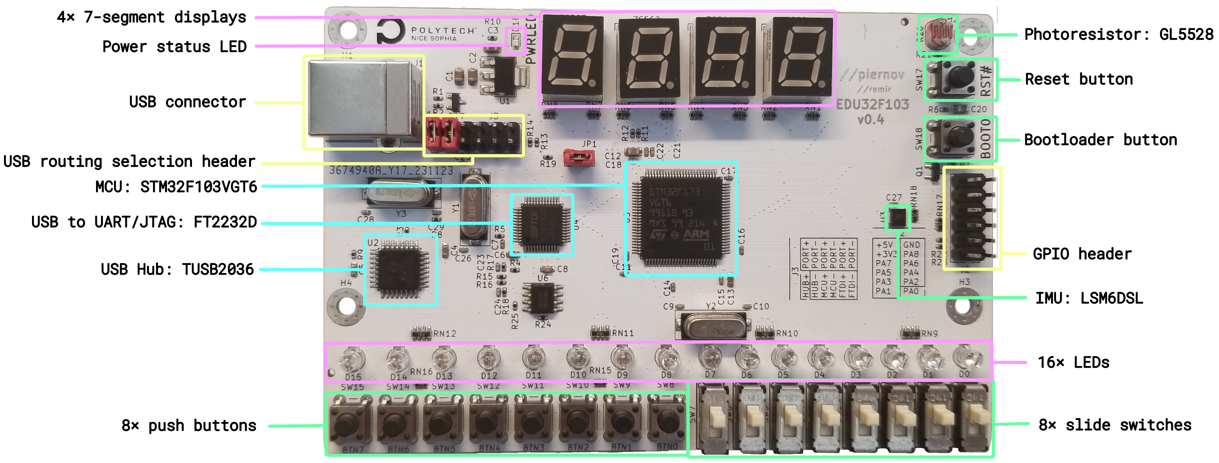


Figure 1: PCB layout

1.1 Description

The EDU32F103 is a development and prototyping board targeted at education and academic use cases. Is it based on an STM32F103-series microcontroller and offers a handful of peripherals and sensors for a quick introduction to the embedded programming world. It is designed with ease-of-use in mind and can be powered and programmed from a single USB Type-B connector using standard open-source tools. All the peripherals are connected directly to the microcontroller, with each peripheral type using its own GPIO port so that using them stays as straightforward as possible. Additionally, the microcontroller can be programmer to be exposed as a USB device to a computer, enabling the switches to be used as a keyboard or a gamepad among other possibilities.

1.2 Features

Peripherals:

- 16× LEDs
- 4× 7-segment displays with decimal point
- 8× push buttons
- 8× slide switches
- Analog photosensor (photoresistor)
- LSM6DSL digital intertial measurement unit with 3-axis accelerometer and 3-axis gyroscope
- GPIO header

Programming and debugging:

- JTAG over USB
- UART over USB

Miscellaneous:

- MCU as USB device

2 Microcontroller

The microcontroller on this board is an STMicroelectronics STM32F103VGT6.

2.1 Characteristics

The main characteristics of this microcontroller are listed below:

- ARM Cortex-M3 core
- 72 MHz maximum frequency
- 1 MiB of Flash
- 96 KiB of RAM

2.2 Clock

The microcontroller's HSE clock input is connected to an 8 MHz quartz with external load capacitors. HSE must be configured in "Crystal/Ceramic resonator" mode instead of "bypass" mode. The HSE crystal oscillator has better stability and accuracy over the internal HSI RC oscillator. Therefore HSE should be preferred over HSI especially when the USB device capabilities are used.

2.3 External reset

The RST# button can be used to reset the microcontroller.

2.4 Bootloader mode

By default, the microcontroller boots from Flash (BOOT0 == 0). When JTAG is enabled in the running firmware, the microcontroller can be reprogrammed on the fly. Otherwise, in order to reprogram the microcontroller when JTAG is disabled or a non-working firmware was programmed previously, the microcontroller must be put in bootloader mode (BOOT0 == 0 && BOOT1 == 0).

In order to do so, press and hold the BOOT0 button, press and release the RST# button, wait for a couple of seconds, then release the BOOT0 button. The same effect can be obtained by holding the BOOT0 button while plugging the USB cable in. LED2 should light up while the BOOT0 button is pressed.

In this mode, the firmware is not loaded from Flash and instead the bootloader is loaded from the boot ROM. The microcontroller can then be reprogrammed through either JTAG or UART.

3 Peripherals

3.1 LEDs

16 LEDs, numbered LED0 to LED15, are directly connected to GPIO port B of the microcontroller. LED0 is connected to PB0, LED1 to PB1 and so on up to LED15 connected to PB15. LEDs are connected in a common anode configuration, with all LED's anodes directly connected to the on-board 3.3V supply, and their cathodes connected to the associated pin on the microcontroller. This means that the microcontroller pins should be controlled in an active-low fashion.

Warning: PB2, PB3 and PB4 are multiplexed with BOOT1, JTDO and NJTRST respectively. BOOT1 is only used when entering bootloader mode so it is inactive under normal operations, meaning that LED2 can be used normally. JTDO and NJTRST are used when the JTAG port of the microcontroller is enabled, in which case LED3 and LED4 cannot be used. In order to use LED3 and LED4, JTAG must be disabled which prevents debugging, as well as programming the microcontroller outside of bootloader mode.

3.2 7-segment displays

4 7-segment displays, numbered 7SEG0 to 7SEG3, are directly connected to GPIO ports D and E of the microcontroller. 7SEG0 is connected to the lower byte of GPIO port D, 7SEG1 is connected to the upper byte of GPIO port D, 7SEG2 is connected to the lower byte of GPIO port E, 7SEG3 is connected to the upper byte of GPIO port E.

Each 7-segment display has 8 segments in total, including the decimal point, named A, B, C, D E, F, G and DP. A is connected to bit 0, B is connected to bit 1 and so on up to DP connected to bit 8 of each byte.

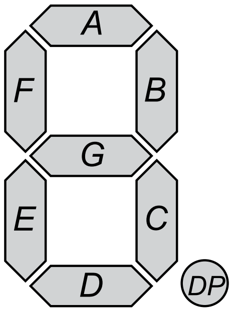


Figure 2: 7-segment display segment names

7-segment displays are connected in a common anode configuration, with all segment's anodes connected to the 5V USB supply, and their cathodes connected to the associated pin on the microcontroller. This means that the microcontroller pins should be controlled in an active-low fashion.

3.3 Slide switches

8 slide switches, numbered SW0 to SW7, are directly connected to the lower byte of GPIO port C of the microcontroller. SW0 is connected to PC0, SW1 is connected to PC1 and so on up to SW7 connected to PC7. Moving the switch in the upper position leaves the GPIO pin floating, while moving it to the lower position pulls the GPIO pin low. The MCU's internal pull-up resistors must be enabled.

3.4 Push buttons

8 push buttons, numbered BTN0 to BTN7, are directly connected to the upper byte of GPIO port C of the microcontroller. BTN0 is connected to PC8, BTN1 is connected to PC9 and so on up to BTN7 connected to PC15 Pressing the button pulls the GPIO pin high, while releasing it pulls the GPIO pin low through external on-board pull-down resistors.

Warning: No hardware debouncing is present for the push buttons.

3.5 Inertial measurement unit

An LSM6DSL Inertial Measurement Unit (IMU) with a 3-axis accelerometer and 3-axis gyroscope is connected to the SPI1 interface on GPIO port A of the microcontroller. Pins CS, SCL, SDA and SDO are connected to PA4/SPI1_NSS, PA5/SPI1_SCK, PA8/SPI1_MOSI and PA6_MISO, respectively. Interrupt pins INT1 and INT2 of the IMU are not connected. Pins SDx and SCx are permanently pulled low.

The IMU sits flat on the top side of the board, refer to Figure 3 and Figure 4 for the orientation and direction of the axes.

Figure 3: IMU direction of detectable acceleration (top view), LSM6DSL datasheet rev 7.

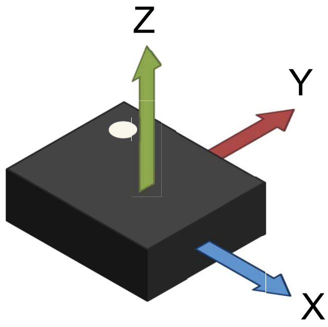
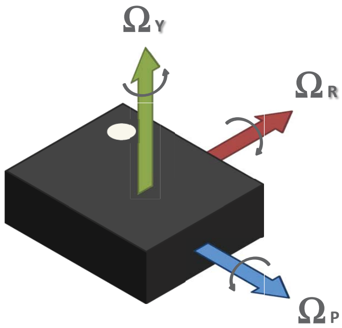


Figure 4: IMU direction of detectable angular rate (top view), LSM6DSL datasheet rev 7.



Warning: PA4, PA5, PA6 and PA7 are shared with the GPIO header. In order to use these pins on the GPIO header and disable the IMU, the resistor network RN18 must be desoldered. To use the IMU and avoid a conflicting signal on the GPIO header, the resistor network RN17 can optionally be desoldered.

3.6 Photoresistor

A GL5528 photoresistor is connected to the ADC1 channel 1 input on GPIO port A of the microcontroller. The photoresistor is the upper resistor of a voltage divider with one pin connected to the on-board 3.3V supply and the other pin connected to PA1. The lower resistor of the voltage divider is a $10k\Omega$ resistor.

Warning: PA1 is shared with the GPIO header. In order to use this pin on the GPIO header and disable the photoresistor, the resistor R22 must be desoldered. To use the photoresistor and avoid a conflicting signal on the GPIO header, the resistor R23 can optionally be desoldered.

3.7 GPIO header

The GPIO header J2 exposes pins PA0 to PA8 from the GPIO port A of the microcontroller, as well as the 5V USB supply, the on-board 3.3V supply and ground. Pins PA0, PA2, PA3 and PA8 are free to use. Pin PA1 is connected to the on-board photoresistor, it may be necessary to desolder R22 in order to disable the photoresistor and use this pin on the GPIO header. Pins PA4, PA5, PA6 and PA7 are connected to the on-board IMU, it is necessary to desolder RN18 in order to disable the IMU and use these pins on the GPIO header.

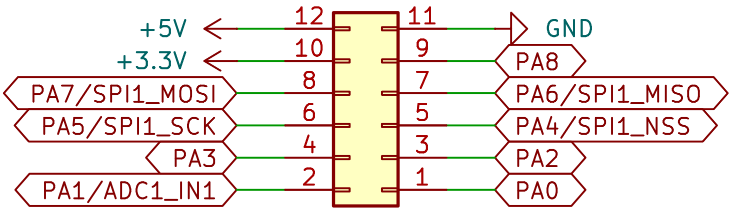


Figure 5: GPIO header pinout

4 USB interface

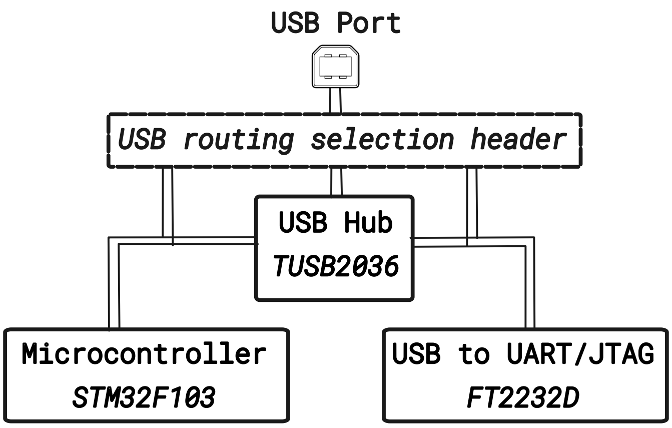


Figure 6: USB architecture

4.1 USB routing selection header

The USB routing selection header J3 is used to chose between connecting the USB data pair from the USB connector (PORT+, PORT-) to either:

- the USB hub (HUB+, HUB-),
- the microcontroller (MCU+, MCU-),
- or the USB to UART/JTAG (FTDI+, FTDI-).

2 jumpers should be used to connect both data lines in a pair to the same interface.

Warning: This selection is only useful when the USB hub U2 is not populated. When the USB hub is populated, the jumpers must be set between PORT+ and HUB+, and between PORT- and HUB-.

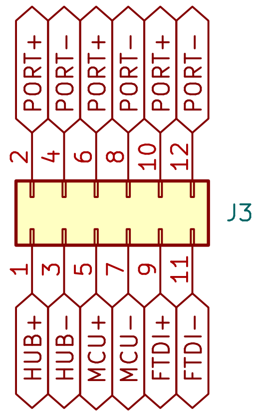


Figure 7: USB routing selection header pinout

4.2 USB hub

The USB hub enables both the microcontroller USB interface and the USB to UART/JTAG interface to be connected to the same USB port and exposed to the computer simultaneously. On some boards, the USB hub U2 may not be populated, in which case the USB routing selection header should be used to choose between either the microcontroller or the USB to UART/JTAG interface.

4.3 JTAG-over-USB

The USB to UART/JTAG interface FT2232D has its first channel connected to the JTAG port of the microcontroller, enabling programming and debugging of the microcontroller through the USB port with standard open-source tool (OpenOCD). 5 JTAG signals from the microcontroller, JTCK, JTDI, JTDO, JTMS and NJTRST, are exposed to the host computer.

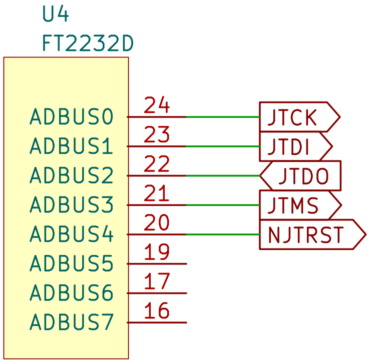


Figure 8: JTAG interface mapping on FT2232D

4.4 UART-over-USB

The USB to UART/JTAG interface FT2232D has its second channel connected to the USART1 interface of the microcontroller on pins PA9 (TX) and PA10 (RX), enabling serial communication with the microcontroller through the USB port. Only the RX and TX signals are exposed to the host computer, PA9 is connected to BDBUS1 and PA10 is connected to BDBUS0.

4.5 MCU USB device

The STM32F103 microcontroller can be used as a USB device exposed on to the host computer through the USB port when programmed appropriately.

5 Microcontroller port mapping

Table 1: Microcontroller port A mapping

Port and pin	Function	Peripheral
PA0	GPIOA0	GPIO header pin 1
PA1	ADC1_IN1, GPIOA1	Photoresistor and GPIO header pin 2
PA2	GPIOA2	GPIO header pin 3
PA3	GPIOA3	GPIO header pin 4
PA4	SPI1_NSS, GPIOA4	IMU pin CS and GPIO header pin 5
PA5	SPI1_SCK, GPIOA5	IMU pin SCL and GPIO header pin 6
PA6	SPI1_MISO, GPIOA6	IMU pin SDA and GPIO header pin 7
PA7	SPI1_MOSI, GPIOA7	IMU pin SDO and GPIO header pin 8
PA8	GPIOA8	GPIO header pin 9
PA9	USART0_TX	FT2232D pin BDBUS1
PA10	USART0_RX	FT2232D pin BDBUS0
PA11	USBFS_DM	USB hub pin DP2, USB routing selection header pin MCU+
PA12	USBFS_DP	USB hub pin DM2, USB routing selection header pin MCU–
PA13	JTMS	FT2232D pin ADBUS3
PA14	JTCK	FT2232D pin ADBUS0
PA15	JTDI	FT2232D pin ADBUS1

Table 2: Microcontroller port B mapping

Port and pin	Function	Peripheral
PB0	GPIOB0	LED0
PB1	GPIOB1	LED1
PB2	GPIOB2, BOOT1	LED2
PB3	GPIOB3, JTDO	LED3, FT2232D pin ADBUS2
PB4	GPIOB4, NJTRST	LED4, FT2232D pin ADBUS4
PB5	GPIOB5	LED5
PB6	GPIOB6	LED6
PB7	GPIOB7	LED7
PB8	GPIOB8	LED8
PB9	GPIOB9	LED9
PB10	GPIOB10	LED10
PB11	GPIOB11	LED11
PB12	GPIOB12	LED12
PB13	GPIOB13	LED13
PB14	GPIOB14	LED14
PB15	GPIOB15	LED15

Table 3: Microcontroller port C mapping

Port and pin	Function	Peripheral
PC0	GPIOC0	SW0
PC1	GPIOC1	SW1
PC2	GPIOC2	SW2
PC3	GPIOC3	SW3
PC4	GPIOC4	SW4
PC5	GPIOC5	SW5
PC6	GPIOC6	SW6
PC7	GPIOC7	SW7
PC8	GPIOC8	BTN0
PC9	GPIOC9	BTN1
PC10	GPIOC10	BTN2
PC11	GPIOC11	BTN3
PC12	GPIOC12	BTN4
PC13	GPIOC13	BTN5
PC14	GPIOC14	BTN6
PC15	GPIOC15	BTN7

Table 4: Microcontroller port D mapping

Port and pin	Function	Peripheral
PD0	GPIOD0	7SEG0_A
PD1	GPIOD1	7SEG0_B
PD2	GPIOD2	7SEG0_C
PD3	GPIOD3	7SEG0_D
PD4	GPIOD4	7SEG0_E
PD5	GPIOD5	7SEG0_F
PD6	GPIOD6	7SEG0_G
PD7	GPIOD7	7SEG0_DP
PD8	GPIOD8	7SEG1_A
PD9	GPIOD9	7SEG1_B
PD10	GPIOD10	7SEG1_C
PD11	GPIOD11	7SEG1_D
PD12	GPIOD12	7SEG1_E
PD13	GPIOD13	7SEG1_F
PD14	GPIOD14	7SEG1_G
PD15	GPIOD15	7SEG1_DP

Table 5: Microcontroller port E mapping

Port and pin	Function	Peripheral
PE0	GPIOE0	7SEG2_A
PE1	GPIOE1	7SEG2_B
PE2	GPIOE2	7SEG2_C
PE3	GPIOE3	7SEG2_D
PE4	GPIOE4	7SEG2_E
PE5	GPIOE5	7SEG2_F
PE6	GPIOE6	7SEG2_G
PE7	GPIOE7	7SEG2_DP
PE8	GPIOE8	7SEG3_A
PE9	GPIOE9	7SEG3_B
PE10	GPIOE10	7SEG3_C
PE11	GPIOE11	7SEG3_D
PE12	GPIOE12	7SEG3_E
PE13	GPIOE13	7SEG3_F
PE14	GPIOE14	7SEG3_G
PE15	GPIOE15	7SEG3_DP

6 Programming environment

6.1 JTAG programming and debugging (OpenOCD)

In order to program the microcontroller through the USB port, the FT2232D 1st channel acts as an USB to JTAG converter. OpenOCD can be used to communicate with the microcontroller, as it supports both the FT2232D and the STM32F103VG.

The following OpenOCD configuration file can be used for this purpose:

Listing 1: edu32f103_openocd.cfg

```
adapter driver ftdi
ftdi vid_pid 0x0403 0x6010
ftdi channel 0
ftdi layout_init 0x0038 0x003b
transport select jtag
adapter speed 200

source [find target/stm32f1x.cfg]
```

For example, to program an ELF file named firmware.elf to the microcontroller's Flash:

```
openocd -f edu32f103_openocd.cfg -c init \
    -c "reset halt; flash write_image erase firmware.elf; reset"
```

The JTAG interface and OpenOCD can also be used for live debugging with GDB.

6.2 UART programming (STM32CubeProgrammer)

As an alternative to JTAG, the microcontroller's internal Flash can be reprogrammed through the UART in bootloader mode. In order to do so, boot the board in bootloader mode (see Section 2.4), and use STM32CubeProgrammer with the following settings:

- Mode: UART
- Port: Select the serial port associated to the 2nd interface (UART) of the FT2232D, e.g., ttyUSB1
- Baud rate: 57600
- Parity: Even
- Data bits: 8
- Stop bits: 1
- Flow control: Off
- RTS: 0
- DTS: 0

Note that this mode does not provide any debugging capabilities.

6.3 Hardware-abstraction layer (STM32Cube)

In order to make the configuration of the microcontroller and accessing its peripherals easier, STMicroelectronics offers STM32CubeMX as a template generation tool and STM32Cube as a hardware-abstraction layer included in this tool.

An STM32CubeMX project with some interfaces already configured is provided. The template source code should then be generated from the STM32CubeMX tool.

Note that STM32CubeMX does not include a toolchain to build the source code. The Arm GNU Toolchain can be used for this purpose.

More information can be found in the `README.md` file provided along with the STM32CubeMX project.

6.4 Real-time operating system (Zephyr)

When a real-time operating system is needed, e.g., for multi-tasking, Zephyr can be used to develop the firmware instead of relying on STM32CubeMX. Zephyr provides its own portable API to interact with the hardware. Zephyr can also be useful even when multi-tasking is not required since it provides many drivers and APIs to reduce the development time.

A demo Zephyr project is provided and includes the required configuration files (Device Tree Source).

More information can be found in the `README.md` file provided along with the Zephyr project.

6.5 Integrated development environment

An integrated development environment (IDE) provides a friendly user interface in order to configure, develop, program and debug a software. IDEs also often provide features for versioning, project management, teamwork, and many other aspects of software engineering.

6.5.1 STM32CubeIDE

STM32CubeIDE is an integrated development environment provided by STMicroelectronics and based upon Eclipse. STM32CubeIDE integrates STM32CubeMX and can also interface seemlessly with the OpenOCD tool.

Please note that Zephyr is not yet integrated into STM32CubeIDE.

6.5.2 Visual Studio Code

Stricly speaking, Visual Studio Code is not an integrated development environment but a source code editor. However, using extensions, several features commonly found in IDEs can be added to VS Code.

For embedded development targeting the EDU32F103 board, we recommend at least the following extensions:

- Microsoft C/C++
- Microsoft Makefile Tools
- Microsoft Serial Monitor
- Marus Cortex-Debug
- Marus Cortex-Debug Device Support Pack STM32F1

With these extensions and an appropriate configuration of VS Code, the board can be programmed and debugged inside VS Code.

VS Code projects for use with STM32CubeMX and Zephyr are provided.

More information can be found in the `README.md` files provided along with these projects.

7 Mechanical characteristics

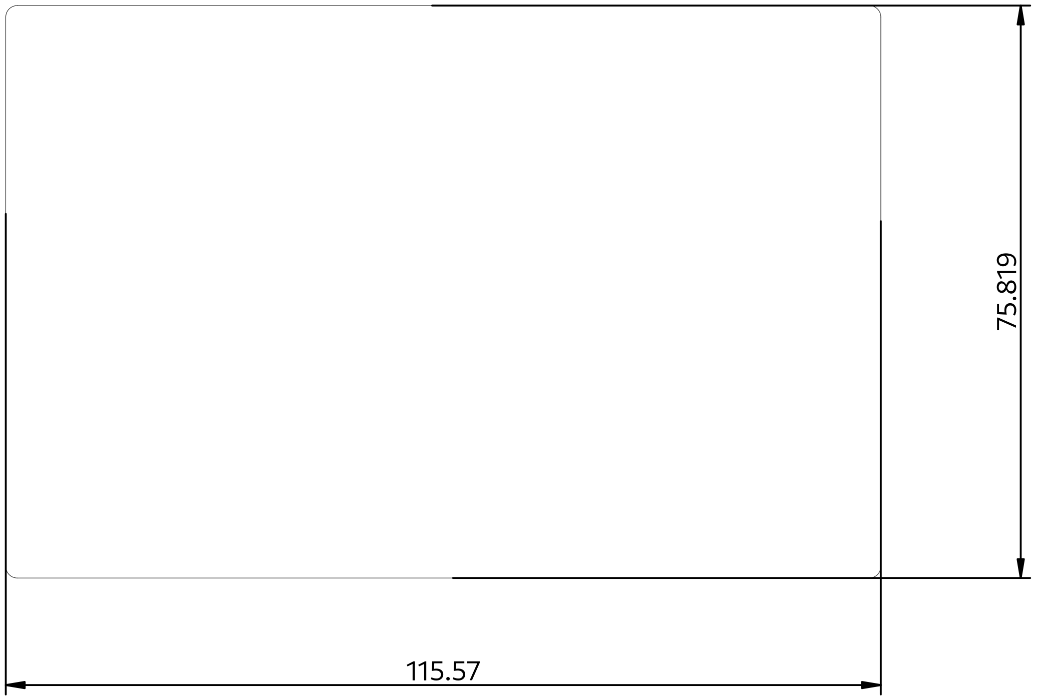


Figure 9: PCB edge cut (mm)

8 Electrical Characteristics

8.1 Absolute maximum ratings

Table 6: Absolute maximum ratings

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDUSB}	USB supply voltage		-0.5		6.0	V
V_{DDMCU}	MCU supply voltage	When supplied externally through JP1	-0.3		4.0	V

8.2 Recommended operating conditions

Table 7: Recommended operating conditions

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{DDUSB}	USB supply voltage			5.0		V
V_{DDMCU}	MCU supply voltage	When supplied externally through JP1		3.3		V

8.3 Supply current characteristics

Table 8: Supply current characteristics

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
I_{DDUSB}	USB supply current	$V_{DDUSB} = 5V$			250	mA
I_{DDMCU}	MCU supply current	When $V_{DDMCU} = 3.3V$ supplied externally through JP1			100	mA

9 Revision History

9.1 Hardware

9.1.1 v0.5

Third production sample.

Table 9: v0.4 to v0.5 hardware changes

Reference designator	Description	Reason
U2	Connect TUSB2036 /GANGED and /BUS-PWR to ground	Fix USB hub stability
U2	Assembled by default	Supply
RN9, RN10, RN11, RN12	Change current-limiting resistors from $1k\Omega$ to $10k\Omega$	Reduce brightness
RN1-RN8	Change current-limiting resistors from $2.2k\Omega$ to $1.5k\Omega$	Increase contrast
R21	Change photoresistor voltage divider lower resistor from $100k\Omega$ to $10k\Omega$	Fix light sensor range
SW0-SW7	Disconnect from 3.3V rail and use internal MCU's pull-up resistors	Fix current leak
R22, R23	Connect photoresistor output to MCU GPIO PA1 instead of PA3	Free USART2 on PA2/PA3

9.1.2 v0.4

Second production sample.

Table 10: v0.3 to v0.4 hardware changes

Reference designator	Description	Reason
U3, RN17, RN18, C27	Added accelerometer LSM6DSL	New feature
R20, R21, R22, R23	Added photoresistor GL5528	New feature
U6, R24, R25	Added EEPROM for FT2232D to customize USB descriptor	New feature
Q1	Added MOSFET for BOOT1 pull-down when BOOT0 pulled-up	Fix bootloader entry
X3, Y2, C9, C10	Changed 8 MHz crystal oscillator X3 to 8 MHz quartz Y2 and load capacitors	Supply and cost reduction
X1, Y3, C28, C29, R8	Changed 48 MHz crystal oscillator X1 to 6 MHz quartz Y3 and load capacitors	Supply and cost reduction
RN9-RN12	Changed 0402 footprint to 0603	Supply and cost reduction
D18	Changed from THT to SMD	Supply and cost reduction
7SEG0-7SEG3	Changed from CA pins 1, 6 to CA pins 3, 6	Supply
SW0-SW7, RN13, RN14	Changed from SPST to SPDT and removed uneeded resistor networks	Supply and user experience
R3, R4, R6, R7, R19	BOM: Corrected resistor values	Fix assembly
R8	BOM: Removed no-stuff resistor	Fix assembly
J2, J3	BOM: Corrected part selection	Fix assembly
N/A	Added Polytech Nice Sophia logo	Branding

9.1.3 v0.3

First production sample.

Table 11: v0.2 to v0.3 hardware changes

Reference designator	Description	Reason
RV1, RV2	Removed uneeded footprints	Fix
N/A	Added backplate	New feature
N/A	Added front glass	New feature
N/A	Created BOM and placement files for automatic assembly	Assembly

9.1.4 v0.2

Second prototype.

Table 12: v0.1 to v0.2 hardware changes

Reference designator	Description	Reason
RV1, RV2	Added footprints	New feature
R1, R2	Removed to fix USB hub	Fix USB hub
R11-R18, C21-C24	Added to fix USB hub	Fix USB hub
X2	Changed footprint	Supply
SW1-SW16	Swapped slide switches and push buttons	User experience
SW1-SW16	Reversed ordering	User experience
SW1-SW16	Renamed to SW0-SW15	User experience
SW8-SW15	Added BTN0-BTN7 silkscreen	User experience
J3	Added header for USB interface selection	New feature
D17, R1, R2	Added USB TVS diode for data lines protection	New feature
R3, R4	Changed footprint from 0603 to 0402	Supply
U3	Removed uneeded footprint	Cleanup
D18, R10	Added power LED	New Feature
H1-H4	Added screw holes	New feature
U5-U8	Reversed ordering	User experience
U5-U8	Renamed to 7SEG0-7SEG3	User experience
N/A	Improved parts placement	Cleanup
N/A	Improved routing and fills	Cleanup

9.1.5 v0.1

First prototype.

9.2 User Manual

9.2.1 v0.2

Second draft for hardware v0.5.

Table 13: v0.1 to v0.2 user manual changes

Section	Description	Reason
3.3	Add note to use internal MCU's pull-up resistors for SW0-SW7 since upper position is floating	v0.5 hardware change
3.4	Clarify that BTN0-BTN7 are pulled low by external on-board pull-up resistors	
3.6, 3.7, 5	Move Photoresistor from ADC1 channel 3 pin PA3 to ADC1 channel 1 pin PA1	v0.5 hardware change
3.6	Change photoresistor voltage divider lower resistor to $10k\Omega$	v0.5 hardware change

9.2.2 v0.1

First draft for hardware v0.4.