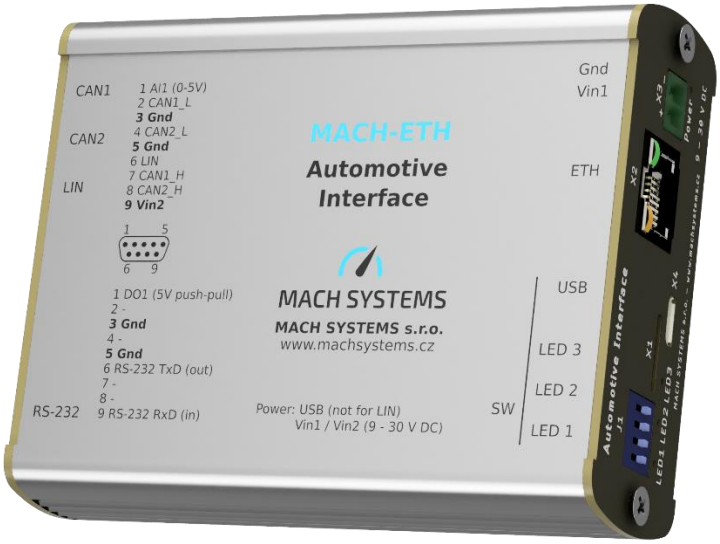
MACH-ETH
User Manual



CHANGES

Date	Description	Created By	Review By
27. 2. 2023	First release	PK	MM

Contents

List of Tables

List of Figures

1 About

This document describes the use of the **MACH-ETH**.

P/N: MACH-ETH

Web: https://www.machsystems.cz/en/products/embedded-networking/gateways-and-bus-converters/mach-eth

Figure 1 MACH-ETH Interface

2 Introduction

The MACH-ETH is an automotive network interface that features one Fast Ethernet port, two CAN FD channels, a LIN channel, a USB and RS-232 port, and multiple I/Os. The communication protocol over Ethernet or USB enables access to CAN(/FD) and LIN bus channels over TCP/IP or virtual COM port respectively.

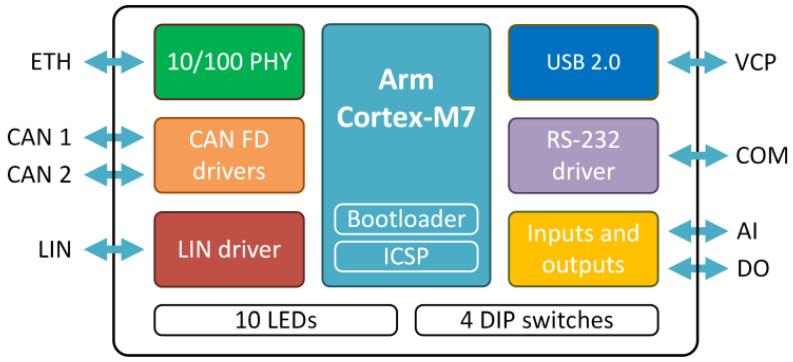


Figure 2 MACH-ETH Block Diagram

The device can act as an interface, a gateway, or bridge between all its channels. An analogue input and a digital output enable the user to interact with external peripherals.

There are **two** options to use the interface:

- A universal “stock firmware” interface for Ethernet-CAN(/FD), Ethernet-LIN, USB-CAN(/FD), USB-LIN with the help of the open communication protocol (see [1]) over Ethernet (both TCP and UDP) or USB (VCP).
- An interface with a user firmware, allowing the user to write his own application on top of the MACH-ETH device.
  Firmware can be developed in C/C++ and can be transferred into the device over USB, ETH or a standard ICSP SWD interface, which also offers code debugging. The device is based on a STM32H7 Arm Cortex-M7 MCU and comes with a free-of-charge IDE, GNU C/C++ compiler, and programming examples.

2.1 Features

- 10/100 Ethernet port
- LIN channel
- RS-232 port
- 2 CAN channels with CAN FD support
- USB 2.0
- Analogue input and digital output
- 10 status LEDs
- 4 DIP switches
- 32-bit Arm Cortex-M7 MCU
- Firmware customizable in C/C++
- Free-of-charge C language SDK
- Free-of-charge IDE and C/C++ compiler
- Firmware upload over USB, ETH or ICSP
- On-board 16 Kbit EEPROM
- Externally or USB-powered
- Table-top use or DIN-rail mount

3 Technical Specification

Communication and Peripherals	
Channels	1 Fast Ethernet (10BASE-T / 100BASE-TX, IEEE 802.3u) 2 CAN-HS (ISO 11898-2) with CAN FD support (ISO 11898-1:2015; CAN 2.0A/B, ISO CAN FD) 1 LIN bus (supports both master and slave; ISO 17987; LIN 2.2a) 1 USB 2.0 CDC (Virtual COM port) 1 RS-232
Integration	Communication protocol for accessing CAN-FD and LIN channels over Ethernet (TCP/IP) and USB (VCP)
User-programming	Free-of-charge IDE and GNU C/C++ compiler (STM32CubeIDE) Programming examples available
Firmware update	over Ethernet, USB, or ICSP (ST-LINK)
Debugging	ST-LINK SWD (a programming header required)

Electrical	
Power	USB-powered over Micro-USB (not for LIN bus) External 9 - 30 V DC power input (polarity and surge protection) over DSUB connector or terminal block
Consumption	100 mA @ 12 V (approx. 1 W)
LEDs	5 Dual-colour LEDs 2 Single-colour LEDs 2 ETH LEDs (RJ-45 connector) 1 Power LED
MCU	STM32H7 (1 MB Flash, 564 KB RAM)
Transceivers	CAN-FD: MCP2562FD LIN: MCP2003B Ethernet: KSZ8041
I/O	1 Analogue input (0-5 V) 1 Digital output (5 V push-pull, 0.5 A, PWM capable)
Mechanical	
Connectors	2 DSUB9M 1 RJ-45 1 Micro-USB 1 Terminal block (2-pin)
Buttons and switches	4 DIP switches
Dimensions (L x W x H)	108 x 82 x 33 mm
Weight	185 g
Operating temperature	-20 to 70 °C
Protection	IP20
Placement	Table (adhesive pads included) DIN-rail mount (sold separately)
Enclosure	Aluminium profile

Table 1 Technical Specification

Figure 3 MACH-ETH Photos

4 Device Description

4.1 Overview

The interface has five connectors, ten LEDs and four DIP switches.

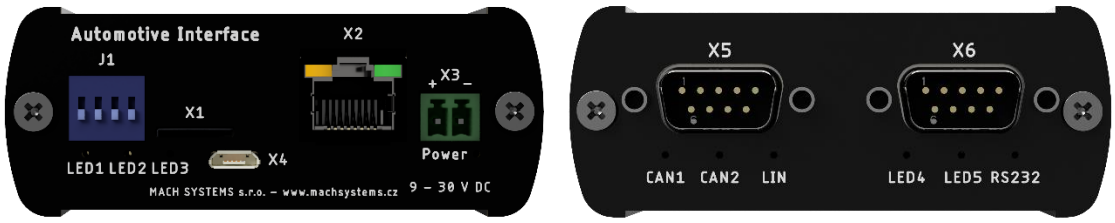


Figure 4 Device Overview

4.2 Power

The MACH ETH can be over microUSB (X4) or externally via a DSUB connector (X5), or a terminal block (X3). All Ground signals are connected.

Note: In order to use the LIN channel, the interface needs to be powered externally (e.g. option 1 or option 2).

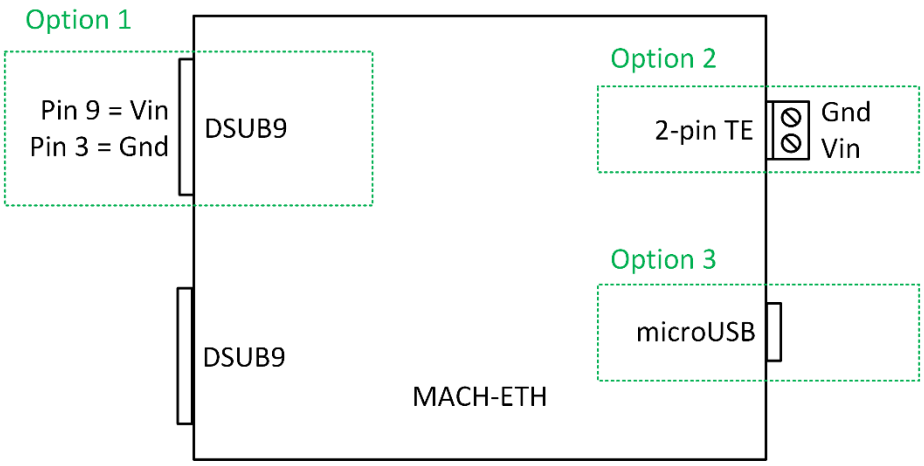


Figure 5 Power Options

When the external power is connected, no power will be drawn from USB.

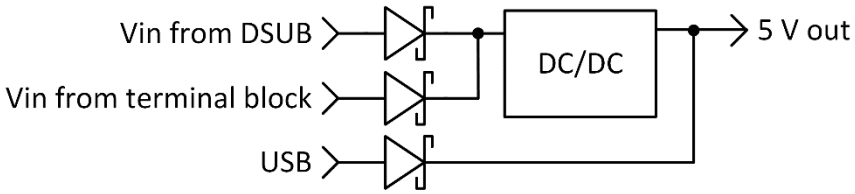


Figure 6 Power Diagram

4.3 LEDs

4.3.1 Power

Green LED is on when the device is powered.

4.3.2 CAN

LED Colour	State	Description
-	Off	Channel stopped
Green	On	Channel started
Green	Blinking	Activity on TX or RX
Red	Blinking	Error frame
Red	On	Error passive

Table 2 CAN Channel LED Description

4.3.3 LIN

LED Colour	State	Description
-	Off	Channel stopped
Green	On	Channel started
Green	Blinking	Activity on TX or RX
Red	Blinking	Slave timeout, checksum error, bus collision, other LIN error

Table 3 LIN Channel LED Description

4.4 Pinout

4.4.1 DSUB connectors

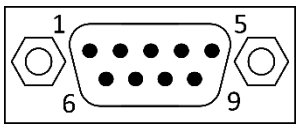
X5 (DSUB9M)	Pin	Name	Note
 Front view	1	AI1	0-5 V
	2	CAN1_L	
	3	Gnd	
	4	CAN2_L	
	5	Gnd	
	6	LIN	
	7	CAN1_H	
	8	CAN2_H	5V push-pull
	9	Vin / Vbat	Power input (used for LIN bus, too)

Table 4 Connector X5 - Pin Assignment

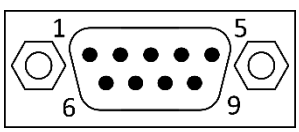
X6 (DSUB9M)	Pin	Name	Note
 Front view	1	DO1	5 V push-pull
	2	-	
	3	Gnd	
	4	-	
	5	Gnd	
	6	RS-232 TxD	out
	7	-	
	8	-	
	9	RS-232 RxD	in

Table 5 Connector X6 - Pin Assignment

4.4.2 Switches

DIP1, DIP2, DIP3 and DIP4 can be used in a user's application.

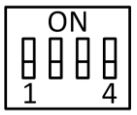
DIP switches
 Side view

Table 6 User DIP Switches

4.5 CAN Bus Termination

There are no internal termination resistors on either CAN channel inside the device. Therefore, a proper termination of the CAN bus is needed, and the user has to make sure the CAN bus is properly terminated at both ends.

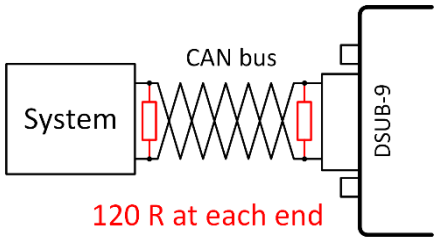


Figure 7 CAN Bus Termination

4.6 Inputs and Outputs

The device features one analogue inputs and one digital outputs (PWM capable).

Signal	Direction	Function	Range	Note
AI1	Input	Analogue / Digital Input	0 - 5 V	
DO1	Output	Push-pull Digital Output	5 V, max. 0.5 A	PWM capable

Table 7 Inputs and Outputs

A load connected to a digital output pin DO1 may draw significant current. In case digital output are used in a user application, it is strongly suggested to power the device from **an external power supply** and not over USB. Otherwise, USB power limits might be exceeded.

4.7 USB

Micro-USB connector (X4) uses the standard pinout, and can be used for firmware upload or in application as a virtual COM port.

4.8 Galvanic Isolation

The device does **not** have any galvanic isolation. The user has to make sure there are no ground loop in his setup

5 Usage

The user can use default firmware to control the device peripherals. The firmware protocol specification is available from [1]. The control message can be sent to the device over Ethernet or USB.

The device's firmware can be also fully developed by the user, and the user has full control over the device's peripherals. For flashing this firmware, the user can make use of the system bootloader which allows for firmware programming over USB or Ethernet bootloader for firmware programming via an internet browser.

The device can be used as CAN FD to CAN bridge, CAN(/FD) to LIN gateway, CAN/LIN to RS-232, data logger, ETH-CAN(/FD), ETH-LIN interface or communication. An analogue input and a digital output enable the user to interact with external peripherals.

5.1 Supplied firmware

With the device following firmware is provided:

- Bootloader
- MachEth (bootloader)
- MachEth (no bootloader)

This firmware is available from [2]. File **Bootloader.bin** contains Ethernet bootloader, which allows upload firmware to device HTTP protocol witch an internet browser. The file **MachEth (bootloader).bin** contains an application with a communication protocol, which allows communication with buses. The Ethernet bootloader file must be flashed into the device before **MachEth (bootloader).bin.** When the **MachEth (bootloader).bin** is flashed to the device through the system bootloader, a user should select another address than 0x08000000 where is bootloader already stored. Recommended address is 0x08020000 for proper bootloader function. When the firmware is flashed over the Ethernet bootloader the correct starting address is selected automatically. If a user wants to use a device without the Ethernet bootloader the **MachEth (no bootloader).bin** should be flashed to the device (this must be done through the system bootloader).

5.2 HTTP interface

Note: the HTTP interface was tested with the Chrome browser. Use of this interface with other browsers is not recommended and it can lead to unpredicted faults.

The default IP address of the device is 192.168.1.100. On the device runs a webserver and after accessing this IP address via browser the web page in Figure 8 is shown. On this page can be found information about the device like its serial number, firmware version hardware info or mac address of the device. This information is static and cannot be changed. There is also information like IP address, subnet mask and TCP port, that can be set by the user on this page and after a click on the save button the device setting is reconfigured. At the bottom of the page are three buttons. One for

device restart, second for accessing the USB bootloader and third for accessing the Ethernet bootloader.

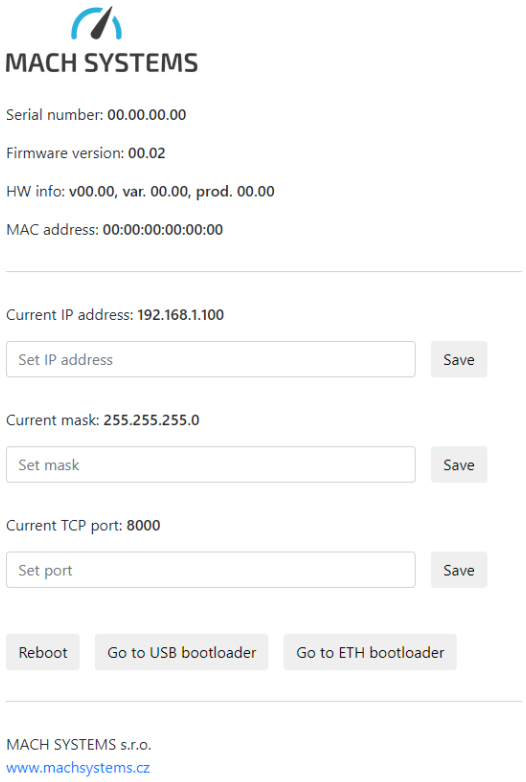


Figure 8 HTTP interface

5.3 System Bootloader

The STM32H7 MCU contains a system bootloader which is pre-programmed in ROM during manufacture. The system bootloader supports USB, it does not support flashing over CAN bus or RS232. If the possibility to upload a firmware over CAN bus is needed, the OpenBLT bootloader described in [2] can be used.

It should be noted that when the device enters the system bootloader whilst USB is connected, the device can then be flashed over USB **only**. If the user wants to flash the device over RS-232, he has to either power the device over external power pins (see Option 1 in 4.2) or he has to make sure USB data lines are not connected. This is the limitation of the system bootloader.

5.4 Ethernet bootloader

With Ethernet bootloader, new firmware can be easily uploaded to the device with only a web browser. No additional software is needed. Recommended web browser for firmware upload is Google Chrome. After entering the Ethernet bootloader (see 5.5) the page in Figure 9 Ethernet bootloader is shown. Users can select a file with firmware and upload it, switch to the system bootloader or go back to the application. The file with firmware must be in binary format (.bin).

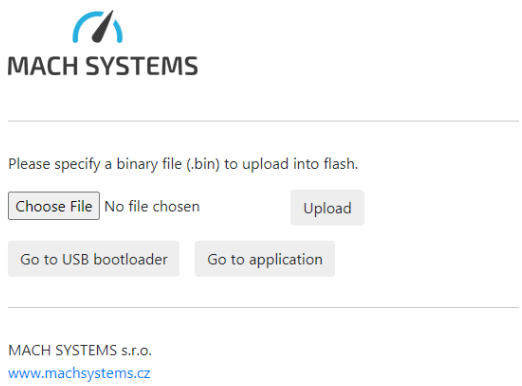


Figure 9 Ethernet bootloader

5.5 How to enter Bootloader

There are four ways to enter the bootloader:

A. Programmatically from application

The user can call *jumpToBootloader*() function whenever he wants to enter the system bootloader. Do not to call the function directly from an interrupt routine. It is suggested to enable at least one way to programmatically enter the system bootloader. In the example application, there are two bootloaders -> two bootloaders addresses. One for the system bootloader second for Ethernet bootloader. An example how is bootloader call (taken from source code example):

```
/* Request going to the bootloader and do not do anything
else – another task will manage this */
BootloaderRequest = pRxPkt->Data[0] == 0 ? 1 : 2;
/* Can not wait infinitely here, as the request is
managed by another task and as this function can be
called from interrupt, the task would never
be scheduled! */
```

In the other task in infinite loop:

```
if (BootloaderRequest == 1)
      jumpToBootloader(SYSTEM_BOOT_ADDR);
else if (BootloaderRequest == 2)

{

      jumpToBootloader(HTTP_BOOT_ADDR);

}
```

B. By hardware

To enter the bootloader open the enclosure to access the boot-enable pads located on the top layer of the PCB. There are two pads one for the system bootloader second for the Ethernet bootloader. The steps for system bootloader:

- Disconnect the USB and the external power supply
- Open the enclosure

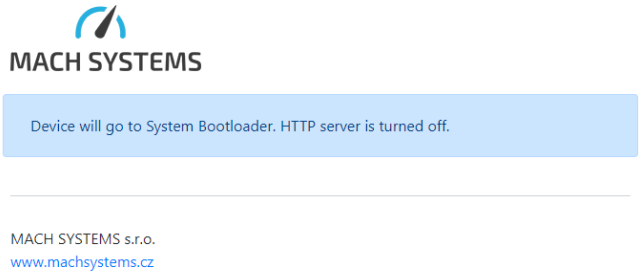
- Short the boot pads together (a pair of tweezers can be used)
- Connect the power supply - either USB or external
- The device will enter the system bootloader
- Release the boot pads
- Firmware can be flashed as described in 5.6
- Close the enclosure

The steps for Ethernet bootloader:

- Disconnect the USB and the external power supply
- Open the enclosure
- Connect the power supply - either USB or external
- Connect the Ethernet cable
- Short the eth boot pads together (a pair of tweezers can be used)
- The device will enter the Ethernet bootloader
- Release the boot pads
- Firmware can be flashed as described in 5.4
- Close the enclosure

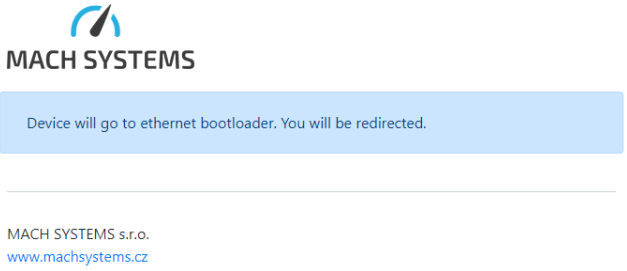
C. With internet browser

- Open Chrome internet browser
- Type IP address (default IP address is 192.168.1.100) of the device in the address bar
- At the bottom of the page, click on the button Go to USB bootloader to enter the system bootloader
  - Window with the short message is shown



  - Firmware can be flashed as described in 5.6

- To enter Ethernet bootloader, click on Go to Ethernet bootloader
  - Window with short message is shown



  - Firmware can be flashed as described in 5.4

D. Send protocol message

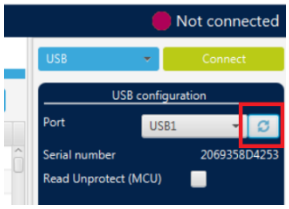
The user sends a protocol message with id 0xFE to enter the bootloader. Selection of bootloader type is done by Data0 byte. After setting this byte to 0, the device will enter the system bootloader. If the byte is set to 1 device will enter the Ethernet bootloader. More details about protocol in MACH-ETH Communication Protocol Specification.

5.6 Firmware Download with Cube Programmer

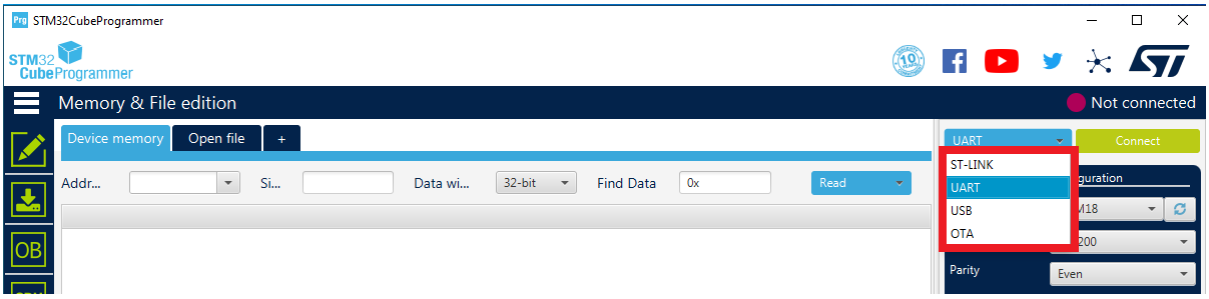
The STM32CubeProgrammer application can be used for flashing firmware over the system bootloader. The application is available from [3].

Steps for uploading a firmware:

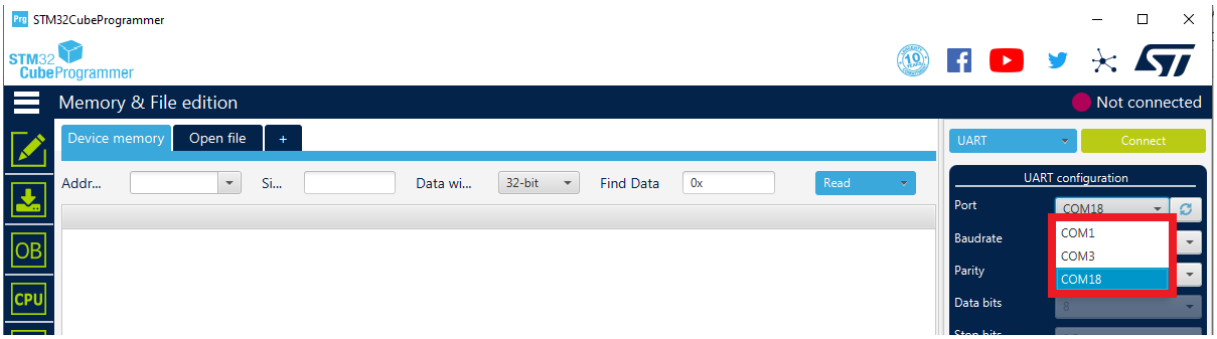
1. Open *STM32CubeProgrammer* application (see download link above)
2. Turn the device off
3. Enter the system bootloader as described in 5.5
4. Connect one of the following ports to the PC:
    a. Micro USB cable (this also powers the device)
    b. RS-232 over pins 6, 9 and 5 (RxD, TxD and Gnd, respectively)
5. Further, the steps are very similar for all the ports. In the *STM32CubeProgrammer*:
    a. Click on the refresh arrows button to see available ports



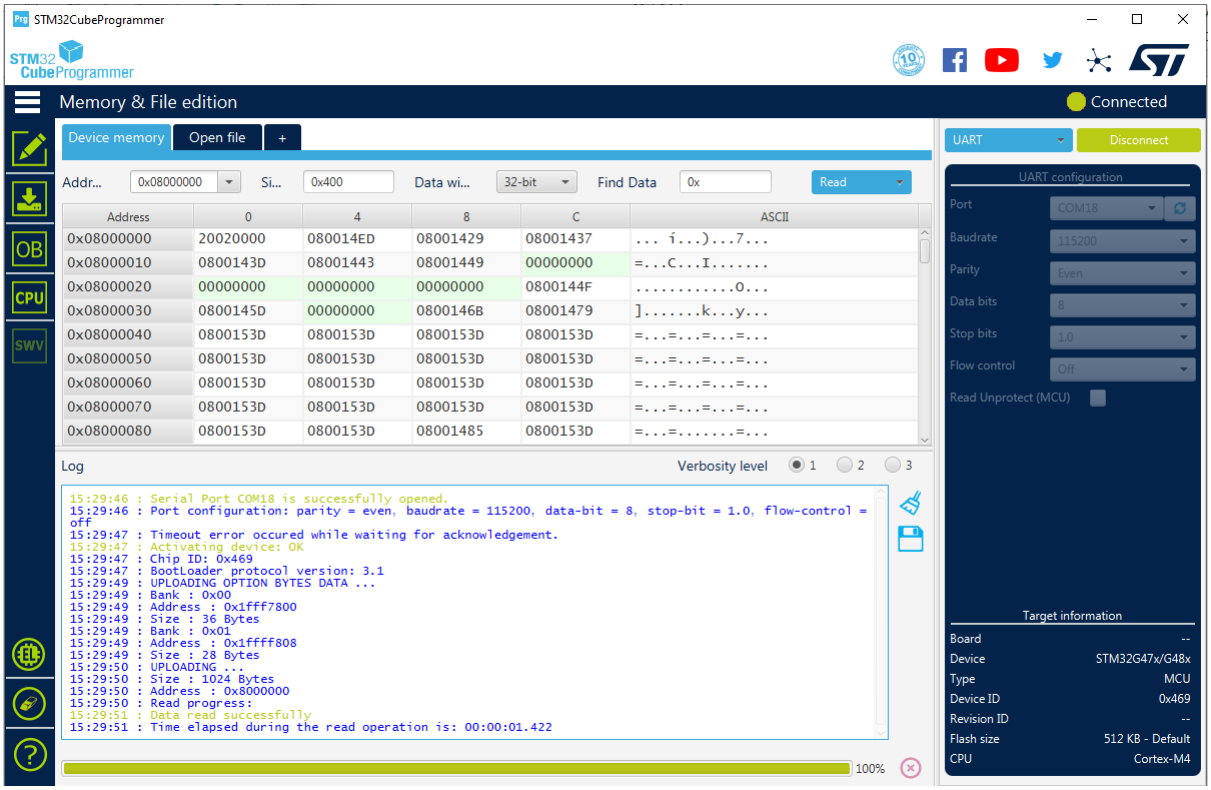
    b. Select the interface (USB or UART)



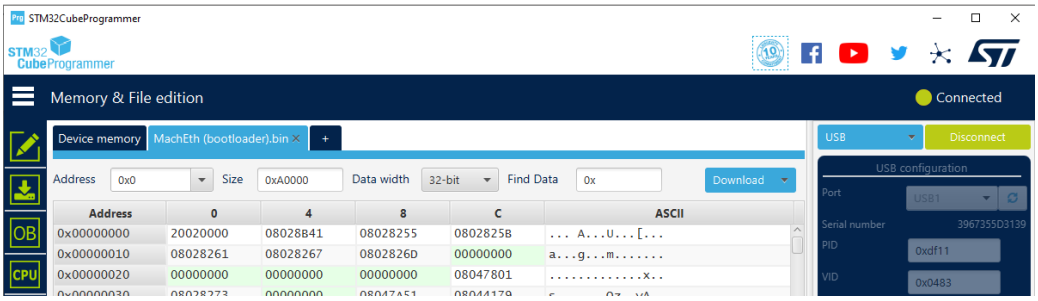
    c. Select the correct port



d. You do not have to change the UART settings provided that the baud rate is lower or equal to 115200. Procedure of connecting to the USB is almost identical.
e. Click connect. You will see screen similar to this one.



6. If everything went well, you are now connected to the bootloader. You can read/write memory, load hex files and edit Option Bytes. You cannot view MCU core window and Serial view window; this is possible only when debugging with connected ST-LINK.
7. In order to flash a new firmware, open an .elf file by „Open file“ button and press „Download“ button. A .hex file can be used also but the Address of 0x8000000 has to be chosen manually. More about selecting right address for specific firmware in 5.



If you want to exit the bootloader, you must restart the device.

5.7 In-Circuit Serial Programming

The ST’s ST-LINK SWD connection can be used for both programming and debugging the code directly on the device. The device’s enclosure will have to be open in order to access ICSP pads.

The SWD signals are available over J5 as depicted in Figure 10.

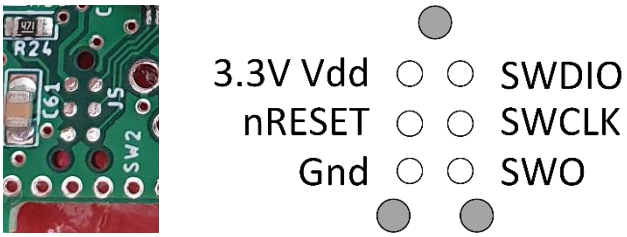


Figure 10 J5 - ST-LINK SWD Connection and Pinout

An ST-LINK v2 or v3 debugger and a TagConnect TC2030-NL header [4] is needed.
p/n: TC2030-CTX-NL-STDC14

Please make sure SWD Interface is selected in STM32CubeIDE project configuration as shown in Figure 11.

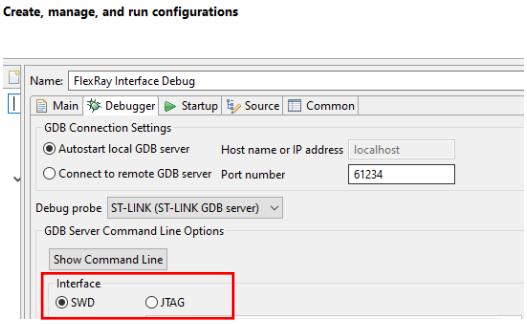


Figure 11 STM32CubeIDE Debugger Configuration

6 Legal Information

6.1 Usage Warning

WARNING FOR ALL USERS

WARNING! - YOUR USE OF THIS DEVICE MUST BE DONE WITH CAUTION AND A FULL UNDERSTANDING OF THE RISKS!

THIS WARNING IS PRESENTED TO INFORM YOU THAT THE OPERATION OF THIS DEVICE MAY BE DANGEROUS. YOUR ACTIONS CAN INFLUENCE THE BEHAVIOR OF A DISTRIBUTED EMBEDDED SYSTEM, AND DEPENDING ON THE APPLICATION, THE CONSEQUENCES OF YOUR IMPROPER ACTIONS COULD CAUSE SERIOUS OPERATIONAL MALFUNCTION, LOSS OF INFORMATION, DAMAGE TO EQUIPMENT, AND PHYSICAL INJURY TO YOURSELF AND OTHERS. A POTENTIALLY HAZARDOUS OPERATING CONDITION IS PRESENT WHEN THE FOLLOWING TWO CONDITIONS ARE CONCURRENTLY TRUE: THE PRODUCT IS PHYSICALLY INTERCONNECTED TO A REAL DISTRIBUTED EMBEDDED SYSTEM; AND THE FUNCTIONS AND OPERATIONS OF THE REAL DISTRIBUTED EMBEDDED SYSTEM ARE CONTROLLABLE OR INFLUENCED BY THE USE OF THE CAN NETWORK. A POTENTIALLY HAZARDOUS OPERATING CONDITION MAY RESULT FROM THE ACTIVITY OR NON-ACTIVITY OF SOME DISTRIBUTED EMBEDDED SYSTEM FUNCTIONS AND OPERATIONS, WHICH MAY RESULT IN SERIOUS PHYSICAL HARM OR DEATH OR CAUSE DAMAGE TO EQUIPMENT, DEVICES, OR THE SURROUNDING ENVIRONMENT.

WITH THIS DEVICE, YOU MAY POTENTIALLY:

- CAUSE A CHANGE IN THE OPERATION OF THE SYSTEM, MODULE, DEVICE, CIRCUIT, OR OUTPUT.
- TURN ON OR ACTIVATE A MODULE, DEVICE, CIRCUIT, OUTPUT, OR FUNCTION.
- TURN OFF OR DEACTIVATE A MODULE, DEVICE, CIRCUIT, OUTPUT, OR FUNCTION.
- INHIBIT, TURN OFF, OR DEACTIVATE NORMAL OPERATION.
- MODIFY THE BEHAVIOR OF A DISTRIBUTED PRODUCT.
- ACTIVATE AN UNINTENDED OPERATION.
- PLACE THE SYSTEM, MODULE, DEVICE, CIRCUIT, OR OUTPUT INTO AN UNINTENDED MODE.

ONLY THOSE PERSONS WHO:

(A) ARE PROPERLY TRAINED AND QUALIFIED WITH RESPECT TO THE USE OF THE DEVICE,

(B) UNDERSTAND THE WARNINGS ABOVE, AND

(C) UNDERSTAND HOW THIS DEVICE INTERACTS WITH AND IMPACTS THE FUNCTION

AND SAFETY OF OTHER PRODUCTS IN A DISTRIBUTED SYSTEM AND THE APPLICATION FOR WHICH THIS DEVICE WILL BE APPLIED, MAY USE THE DEVICE.

PLEASE NOTE THAT YOU CAN INTEGRATE THIS PRODUCT AS A SUBSYSTEM INTO HIGHER-LEVEL SYSTEMS. IN CASE YOU DO SO, MACH SYSTEMS s.r.o. HEREBY DECLARES THAT MACH SYSTEMS s.r.o.'s WARRANTY SHALL BE LIMITED TO THE CORRECTION OF DEFECTS, AND MACH SYSTEMS s.r.o. HEREBY EXPRESSLY DISCLAIMS ANY LIABILITY OVER AND ABOVE THE REFUNDING OF THE PRICE PAID FOR THIS DEVICE, SINCE MACH SYSTEMS s.r.o. DOES NOT HAVE ANY INFLUENCE ON THE IMPLEMENTATIONS OF THE HIGHER-LEVEL SYSTEM, WHICH MAY BE DEFECTIVE.

6.2 Disposal and Recycling Information

When this product reaches its end of life, please dispose of it according to your local environmental laws and guidelines.

6.3 Declaration of Conformity

MACH SYSTEMS

EU Declaration of Conformity (DoC)

We

Company Name	MACH SYSTEMS s.r.o.		
Postal Address	Pocernicka 272/96	**City**	Prague
Postcode	108 00	**Country**	Czech Republic

declare that the DoC is issued under our sole responsibility and belongs to the following products:

MACH-ETH Automotive Interface
MACH-FRAY Automotive Interface

Objects of the declaration:

Product	Product Number
MACH-ETH Automotive Interface	MACH-ETH
MACH-FRAY Automotive Interface	MACH-FRAY

The objects of the declaration described above is in conformity with the relevant Union harmonisation legislation:

2014/30/EU - EMC Directive
2011/65/EU - RoHS (recast)

The following harmonised standards and technical specifications have been applied:

EN 55032	EN 61000-4-2
EN 63000	EN 61000-4-4

Signed for and on behalf of: MACH SYSTEMS s.r.o.

Place of issue: Prague, Czech Republic

Date of issue: February 24th 2023

Signature: 

Name, function: Miroslav Machacek, Managing Director

MACH SYSTEMS s.r.o.
www.machsystems.cz

6.4 Patents, Copyrights and Trademarks

7 References

[1] “MACH-ETH Communication Protocol Specification,” [Online]. Available: https://www.machsystems.cz/support/MACH-ETH - Gateway Protocol Specification_latest.pdf.

[2] “MACH SYSTEMS - Support Page,” [Online]. Available: https://www.machsystems.cz/en/support.

[3] “STM32CubeProgrammer Web Site,” [Online]. Available: https://www.st.com/en/development-tools/stm32cubeprog.html.

[4] “TagConnect TC2030-IDC-NL,” [Online]. Available: https://www.tag-connect.com/product/tc2030-ctx-nl-stdc14-for-use-with-stm32-processors-with-stlink-v3.

8 Ordering Information

Product Number	Description
MACH-ETH	MACH-ETH device
DIN-BRACKET-UNI	A universal bracket for mounting any enclosure on a DIN rail
MACH-ETH-NET-SDK	.NET C# SDK API (DLL) for accessing CAN-FD and LIN channels over Ethernet (TCP/IP) and USB (VCP)

Table 8 Ordering Numbers

9 Contact

MACH SYSTEMS s.r.o.

www.machsystems.cz

info@machsystems.cz

Czech Republic

MACH SYSTEMS

Company registration: 29413893

EU VAT number: CZ29413893