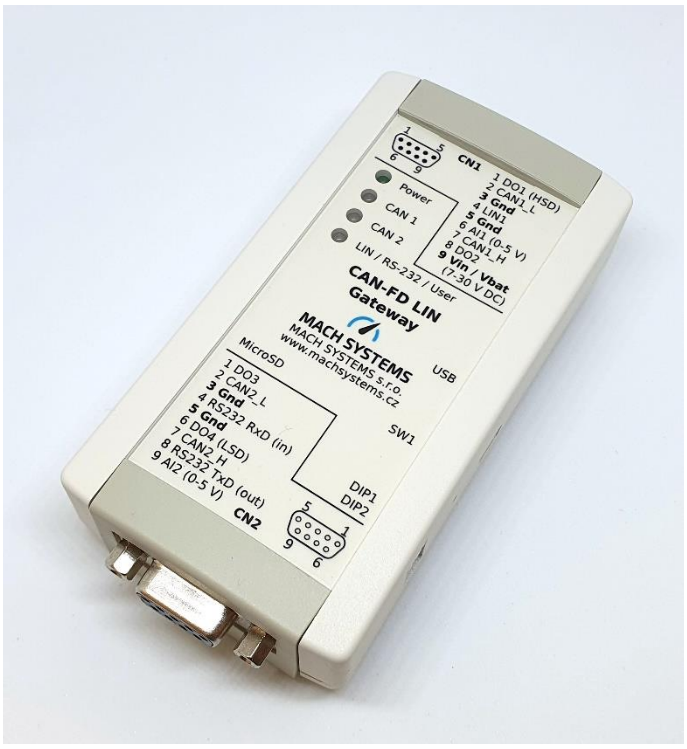
CAN FD LIN Gateway

CAN Bootloader Application Note



CHANGES

Date	Description	Created By	Review By
25.1.2021	Initial Release	KH, VB, MM	MM

Contents

List of Tables

List of Figures

1. Introduction

The **CAN FD LIN Gateway** is a user-programmable device. The firmware can be developed in the STM32CubeIDE and transferred into the device, and the user can upload the firmware over USB, RS-232, or the CAN bus.

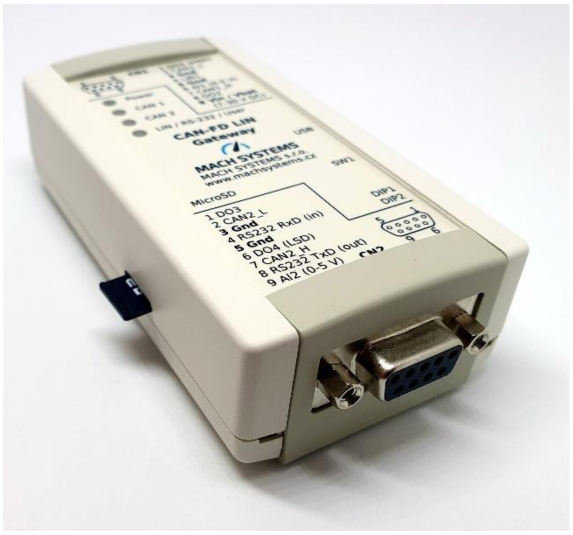


Figure 1 Front side

The STM32G4 MCU contains a system bootloader – please refer to [1] - which is pre-programmed in ROM during manufacture. The system bootloader supports USB and RS-232, it does not support flashing over CAN bus. If the possibility to upload a firmware over CAN bus is needed, the OpenBLT bootloader described in this application note can be used.

2. OpenBLT Bootloader

Because the built-in bootloader on STM32G4 devices does not support flashing over CAN bus the user can use the **OpenBLT bootloader**.

It is an open-source bootloader which has been ported onto the STM32G4 family. It can be used via both CAN channels and RS232/UART interfaces[1]. You can upload your firmware using the `MicroBoot.exe` utility that is contained within the OpenBLT distribution [2].

[1] For switching between CAN channels and RS232/UART, you must change one line in the `blt_conf.h` file in the **bootloader** source:

```
/** \brief Select the desired CAN peripheral as a zero based index. */
#define BOOT_COM_CAN_CHANNEL_INDEX      (0)
/** \brief Select the desired UART peripheral as a zero based index. */
#define BOOT_COM_RS232_CHANNEL_INDEX     (0)  /* 0 is RS232 (USART1), 1 is DEBUG UART (USART2) */
```

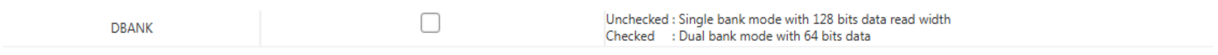
Value	Meaning
BOOT_COM_RS232_CHANNEL_INDEX = 0	Using RS232

When the OpenBTL bootloader is flashed to the MCU, after power-up, there is 500 ms wait for the connection from the host PC. If no connection is present after this timeout, the user application is executed (if was flashed previously). Presence of the user program is checked using a checksum, that is stored as a last item in the program's vector table.

In order to make your STM32CubeIDE project suitable for the OpenBTL bootloader, it has to be slightly modified.

2.1. CAN Bootloader Compilation

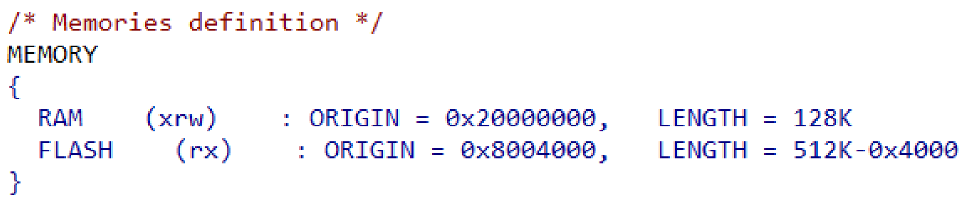
1. You should ensure that you have set DBANK bit in the User Configuration option bytes to 0. It means that we are using the single bank mode. If the DBANK bit is in 1, all works correctly until you want to use upper 256 kilobytes of the flash memory; then the firmware upload won't work. The setting should look like this in the STM32CubeProgrammer (in the menu OB -> User Configuration):



DBANK ☐ Unchecked : Single bank mode with 128 bits data read width
Checked : Dual bank mode with 64 bits data

Figure 2 DBANK option

2. Next, in your application flash linker script (e.g. `STM32G483RETX_FLASH.ld`), set beginning of the flash region to 0x8004000 and decrease the used size by 0x4000 (16 KB). This is because first 16 KB of flash is reserved for the bootloader. The memory definition will look something like this:



```
/* Memories definition */
MEMORY
{
  RAM    (xrw)    : ORIGIN = 0x20000000,   LENGTH = 128K
  FLASH    (rx)    : ORIGIN = 0x8004000,   LENGTH = 512K-0x4000
}
```

Figure 3 Declaration of memories

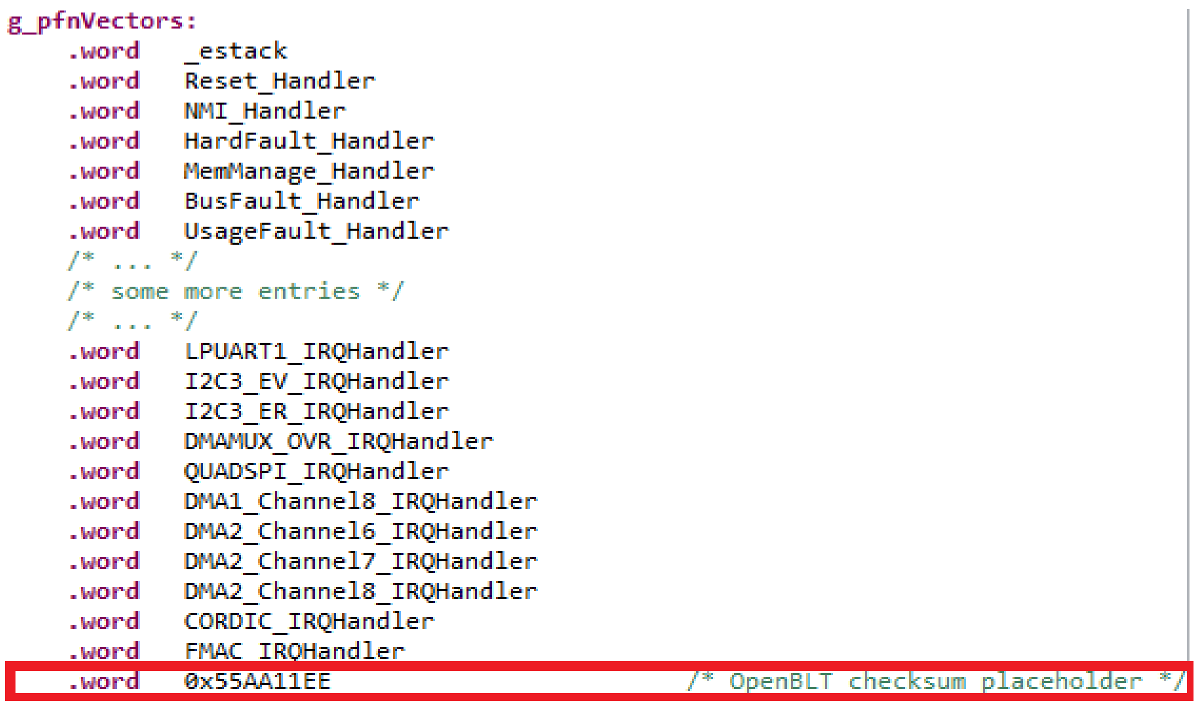
Code for copy&paste:

`FLASH    (rx)    : ORIGIN = 0x8004000,   LENGTH = 512K-0x4000`

Table 1 Copy&paste code (see Figure 3)

3. Make dummy entry in the end of your interrupt vector table. Definition of this is in the startup assembly file named `startup_stm32g483retx.s` located in Core -> Startup in the IDE. The table will (illustratively) look like this (you are adding the last line only):

BOOT_COM_RS232_CHANNEL_INDEX = 1	Using Debug Uart
BOOT_COM_CAN_CHANNEL_INDEX = 0	Using CAN1
BOOT_COM_CAN_CHANNEL_INDEX = 1	Using CAN2



```
g_pfnVectors:
    .word   _estack
    .word   Reset_Handler
    .word   NMI_Handler
    .word   HardFault_Handler
    .word   MemManage_Handler
    .word   BusFault_Handler
    .word   UsageFault_Handler
    /* ... */
    /* some more entries */
    /* ... */
    .word   LPUART1_IRQHandler
    .word   I2C3_EV_IRQHandler
    .word   I2C3_ER_IRQHandler
    .word   DMAMUX_OVR_IRQHandler
    .word   QUADSPI_IRQHandler
    .word   DMA1_Channel8_IRQHandler
    .word   DMA2_Channel6_IRQHandler
    .word   DMA2_Channel7_IRQHandler
    .word   DMA2_Channel8_IRQHandler
    .word   CORDIC_IRQHandler
    .word   FMAC_IRQHandler
    .word   0x55AA11EE                          /* OpenBLT checksum placeholder */
```

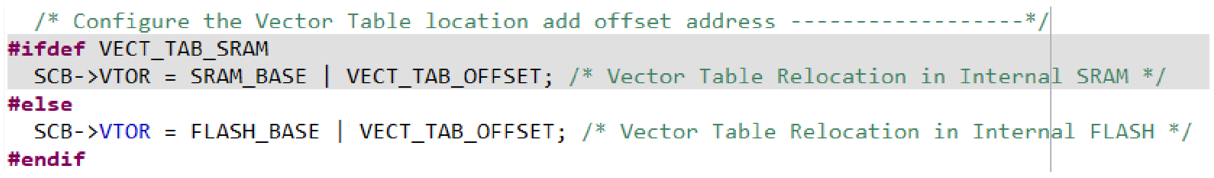
Figure 4 Dummy entry in vector table

Code for copy&paste:

```
.word 0x55AA11EE                    /* OpenBLT checksum placeholder */
```

Table 2 Copy&paste code (see Figure 4)

4. If you are using `SystemInit()` function from the STM library, you must **comment out** the lines with the relocation of the vector table (the base address is modified by the bootloader before jumping to the user application, so omitting this step would break interrupts). So, those lines in the file `system_stm32g4xx.c` must be **commented out**:



```
  /* Configure the Vector Table location add offset address ------------------*/
#ifdef VECT_TAB_SRAM
  SCB->VTOR = SRAM_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal SRAM */
#else
  SCB->VTOR = FLASH_BASE | VECT_TAB_OFFSET; /* Vector Table Relocation in Internal FLASH */
#endif
```

Figure 5 Vector table configuration

5. The application is now done. Last step is to convert the default output of the IDE (.elf) to the S-RECORD (.srec) format, that is used by the `MicroBoot` utility supplied with the OpenBLT distribution (as mentioned in the first paragraph). You can do this in the IDE:
   a. Select your project and click Project -> Properties.

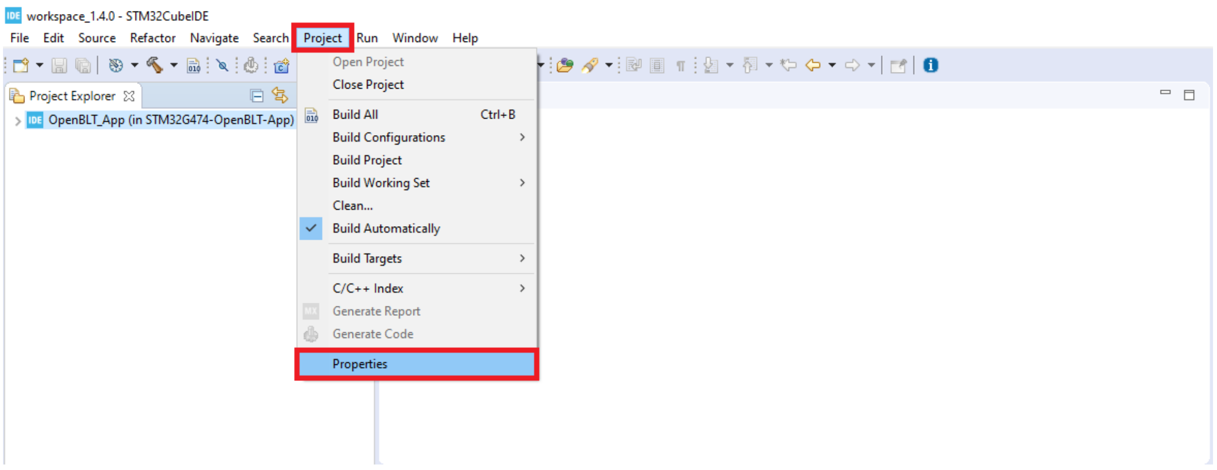
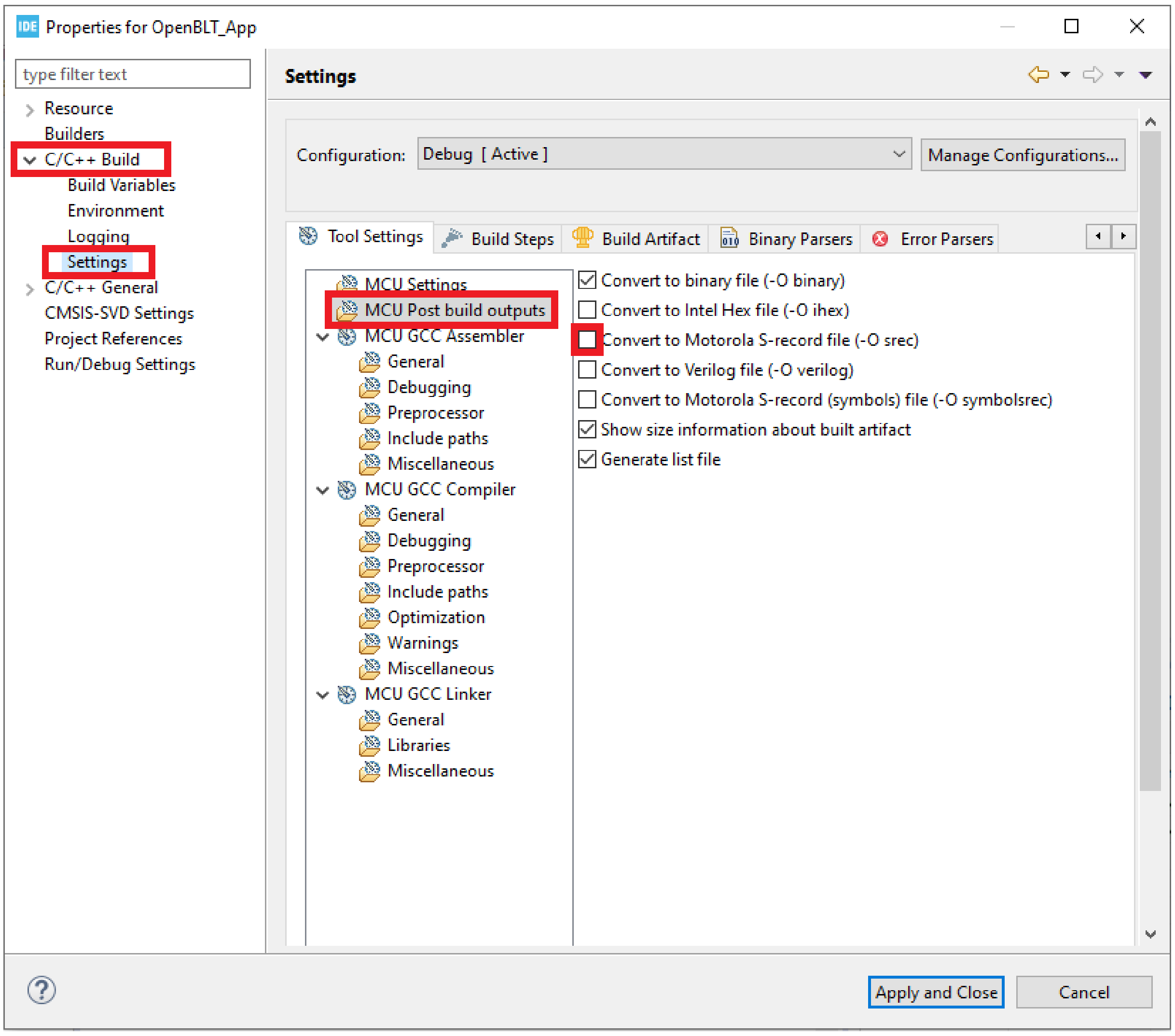


Figure 6 Route to the conversion

b. In the window that opens, click C/C++ Build -> Settings -> MCU Post build outputs and check "Convert to Motorola S-record file (-O srec)"



- [✓] Convert to binary file (-O binary)
- [ ] Convert to Intel Hex file (-O ihex)
- [ ] Convert to Motorola S-record file (-O srec)
- [ ] Convert to Verilog file (-O verilog)
- [ ] Convert to Motorola S-record (symbols) file (-O symbolsrec)
- [✓] Show size information about built artifact
- [✓] Generate list file

Figure 7 Settings of conversion

One of the output files in your build folder (Debug/ or Release/) should now be in the S-RECORD format. In older versions of the IDE, there was a bug where the file was not created properly and it was in the intel HEX format. You would recognize the problem in the next step, when the `MicroBoot` utility won't recognize the file and will refuse to upload it. In the case that you have this problem, refer to Appendix A.

Now the CAN bootloader is compiled and you can flash it into the device as described in User Manual [1].

2.2. CAN Bootloader Usage

Steps to upload the user's firmware over CAN bus:

1. Open `MicroBoot.exe`, which you downloaded with the OpenBLT.
2. In settings select the desired interface (RS232 or CAN).

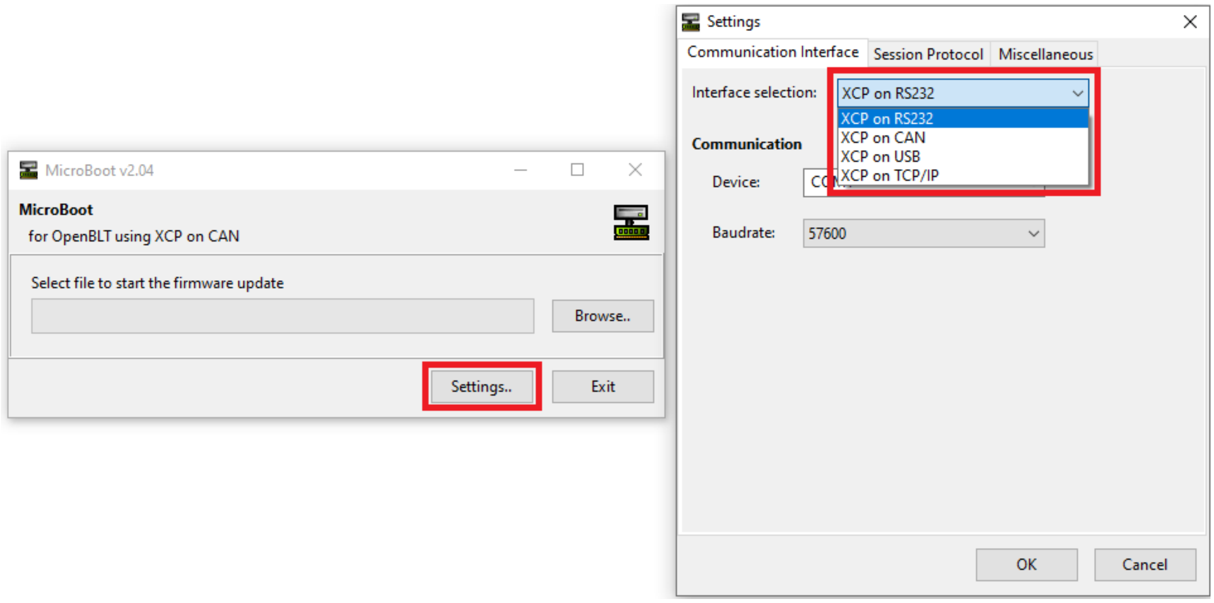
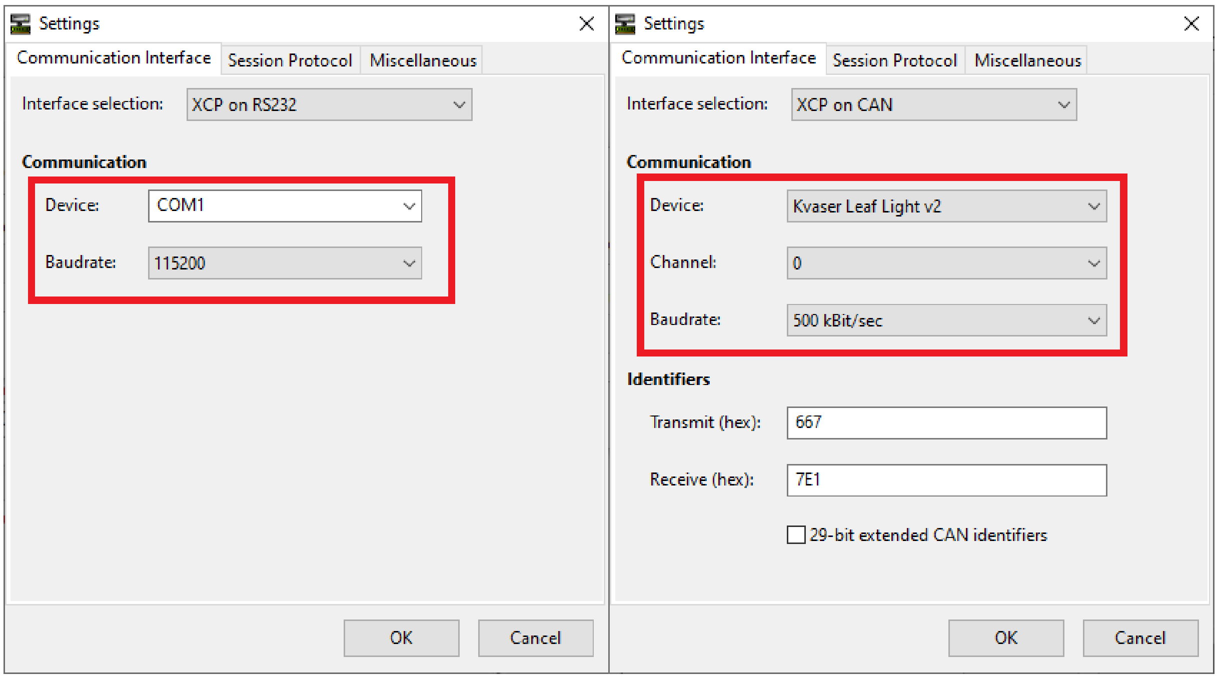


Figure 8 MicroBoot application

3. For RS232, set your desired communication port and baud rate 115200. For CAN, select your communication device and its channel. Baud rate is 500 kBit/s. See the picture below for reference


Settings
Communication Interface | Session Protocol | Miscellaneous

Interface selection: XCP on RS232

Communication

Device: COM1

Baudrate: 115200

OK | Cancel

Settings
Communication Interface | Session Protocol | Miscellaneous

Interface selection: XCP on CAN

Communication

Device: Kvaser Leaf Light v2

Channel: 0

Baudrate: 500 kBit/sec

Identifiers

Transmit (hex): 667

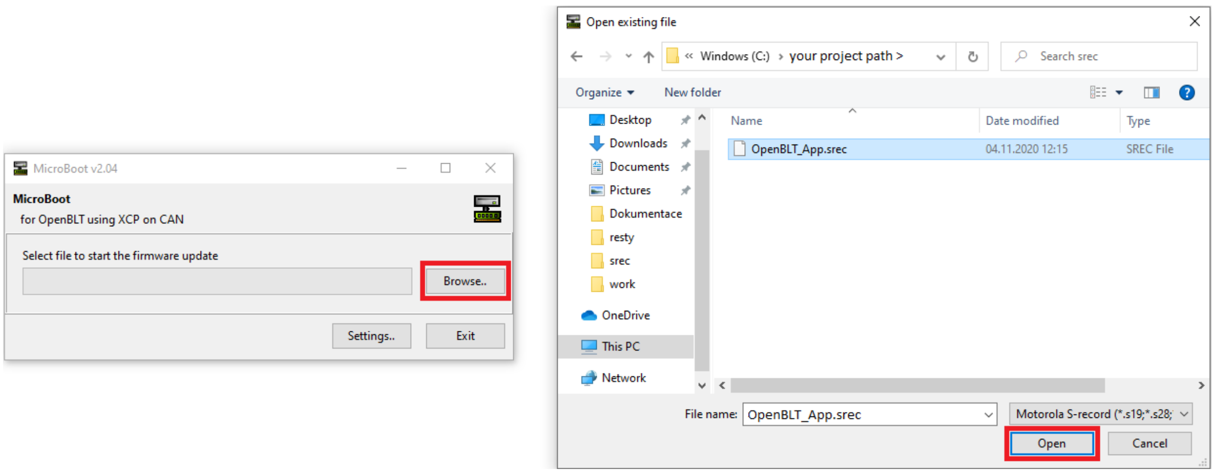
Receive (hex): 7E1

☐ 29-bit extended CAN identifiers

OK | Cancel

Figure 9 Communication Settings

4. Final step: Click Browse and find the *srecord* file created earlier. After pressing Open, the upload should begin automatically. If the connection cannot begin, try restarting the device.


MicroBoot v2.04

MicroBoot
for OpenBLT using XCP on CAN

Select file to start the firmware update

Browse..

Settings.. | Exit

Open existing file

« Windows (C:) › your project path ›

Search srec

Organize ▾ New folder

Name	Date modified	Type
OpenBLT_App.srec	04.11.2020 12:15	SREC File

Desktop, Downloads, Documents, Pictures, Dokumentace, resty, srec, work, OneDrive, This PC, Network

File name: OpenBLT_App.srec
Motorola S-record (*.s19;*.s28;

Open | Cancel

Figure 10 Upload to the device

Once the progress bar finishes and the `MicroBoot` closes, the firmware has been flashed. If everything is ok, the bootloader will automatically execute the uploaded application.

Appendix A - Alternative Way to Generate SREC File

In the older versions of STM32CubeIDE there was a bug which caused that the output in S-RECORD format was not generated properly. However, open-source tools for converting files from HEX to SREC can be used instead.

1. First thing you must do is converting the default output of the IDE (.elf) to the HEX file. It can be done by pasting following code into the Project Properties -> C/C++ Build -> Settings -> Build steps -> Post-build steps -> Command. The output will be named the same as your project name and will be saved in the IDE build folder (Debug/ or Release/). The command:

```
arm-none-eabi-objcopy -O ihex ${"ProjName"}.elf ${"ProjName"}.hex
```

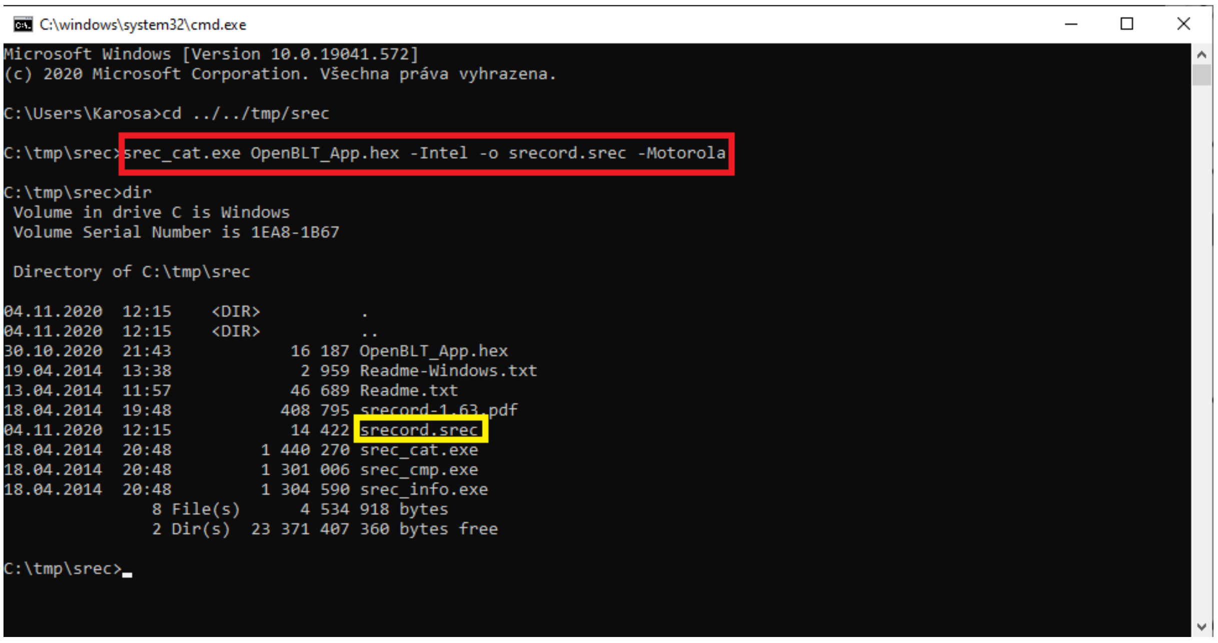
Table 3 Output - HEX conversion command

2. Second step step is converting the created HEX file to the SRECORD format. For this conversion, you must download an external tool.
    a. Download the zip from https://sourceforge.net/projects/srecord/files/ and extract it.
    b. Copy the created HEX from the project folder to the folder, where you extracted the zip.
    c. In `cmd.exe`, change directory to the aforementioned extraction folder.
    d. Run the command

```
srec_cat.exe "filename.hex" -Intel -o srecord.srec -Motorola
```

Table 4 HEX - SRECORD conversion command

Where `filename.hex` is the name of your generated hex file and output will be `srecord.srec`. In the picture below, there is example run – you can see the conversion command in red rectangle and the actual output file in the yellow one.



C:\windows\system32\cmd.exe

```
Microsoft Windows [Version 10.0.19041.572]
(c) 2020 Microsoft Corporation. Všechna práva vyhrazena.

C:\Users\Karosa>cd ../../tmp/srec

C:\tmp\srec>srec_cat.exe OpenBLT_App.hex -Intel -o srecord.srec -Motorola

C:\tmp\srec>dir
 Volume in drive C is Windows
 Volume Serial Number is 1EA8-1B67

 Directory of C:\tmp\srec

04.11.2020  12:15    <DIR>          .
04.11.2020  12:15    <DIR>          ..
30.10.2020  21:43            16 187 OpenBLT_App.hex
19.04.2014  13:38             2 959 Readme-Windows.txt
13.04.2014  11:57            46 689 Readme.txt
18.04.2014  19:48           408 795 srecord-1.63.pdf
04.11.2020  12:15            14 422 srecord.srec
18.04.2014  20:48         1 440 270 srec_cat.exe
18.04.2014  20:48         1 301 006 srec_cmp.exe
18.04.2014  20:48         1 304 590 srec_info.exe
               8 File(s)      4 534 918 bytes
               2 Dir(s)  23 371 407 360 bytes free

C:\tmp\srec>_
```

Figure 11 HEX - SRECORD conversion

You can use the resulting file for upload using the bootloader.

3. References

[1] "CAN FD LIN Gateway User Manual," [Online]. Available: https://www.machsystems.cz/wp-content/uploads/2021/01/CAN-FD-LIN-Gateway-User-Manual.pdf.

[2] "OpenBTL Bootloader Web Site," [Online]. Available: https://www.feaser.com/openblt/doku.php?id=download.

[3] "CAN Bootloader Binaries," [Online]. Available: https://www.machsystems.cz/wp-content/uploads/2021/01/CANFD-LIN-Gateway-OpenBTL-CAN-Bootloader-v1.0.zip.

4. Contact

MACH SYSTEMS s.r.o.

www.machsystems.cz

info@machsystems.cz

Czech Republic

Company registration: 29413893

EU VAT number: CZ29413893