DC11307
Driver 1.02
October 2021
A. Colombo
R. Ertolupi

Interfacing FlashRunner High-Speed with CAN and CAN-FD networks

1. Introduction

This document aims to explain how a CAN or CAN-FD network can be interfaced to FlashRunner High-Speed using its dedicated Active Module, from both hardware and software points of view.

CAN, which stands for Controller Area Network, is an automotive protocol used worldwide because of its robustness and flexibility. The development of this protocol started between 1980 and 1990 at Bosch and then it has become a standard for on-board diagnostics (OBD). Since the original CAN has some limitations in terms of bandwidth and performance, recently a new standard has been introduced in the market: the CAN-FD (Controller Area Network Flexible Data-Rate), which can reach a data rate up to 12 Mbps.

Using the solution developed by SMH Technologies, customers can now interface both CAN and CAN-FD networks with a single hardware solution. This system is designed to satisfy an increasing market demand for programming boards at the end-of-line stage where the case is mounted to the boards and only the CAN bus is accessible. Moreover, this system can be used to flash those devices that only have the CAN or CAN-FD bus as flashing port.
You can download the latest version of this document from this static link: Interfacing FlashRunner High-Speed with CAN and CAN-FD networks.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

2. Contents

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

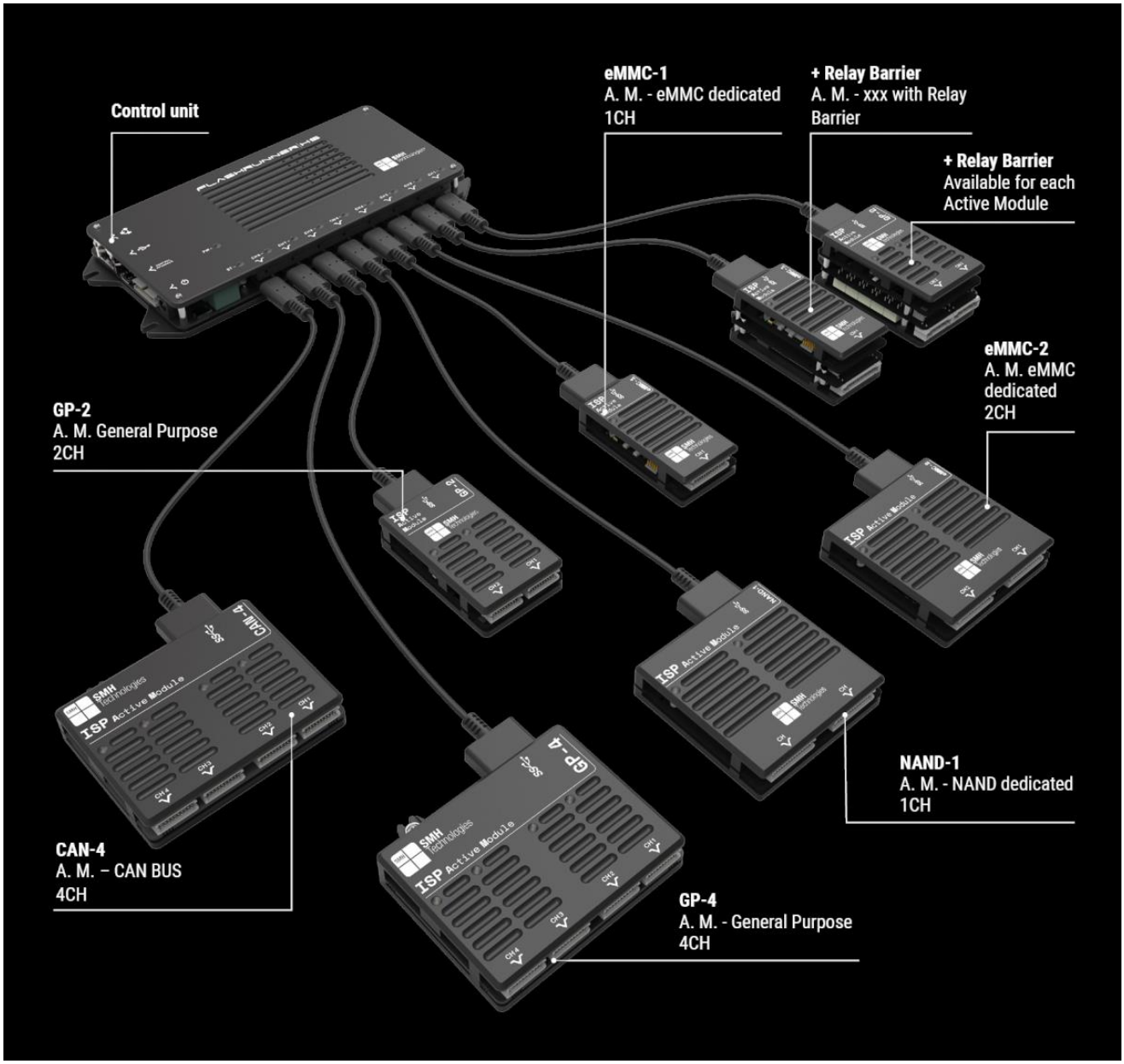
D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

3. Hardware Setup

Before going deep into the CAN specifications, let us spend some words to describe FlashRunner High-Speed and its unique characteristics. FlashRunner High-Speed combines high-level programming performances and high modularity to obtain a Multi-end programming solution that fits the needs of Pre-Programming and In-System Programming equipment. The HS Control Unit is the central management that coordinates different technologies for each peripheral Active Module: this organization reflects the Industry 4.0 concept, where a central intelligence creates smart networking and parallel independent process management, reaching high-quality levels and optimizing the production process. More info about FlashRunner High-Speed can be found here.



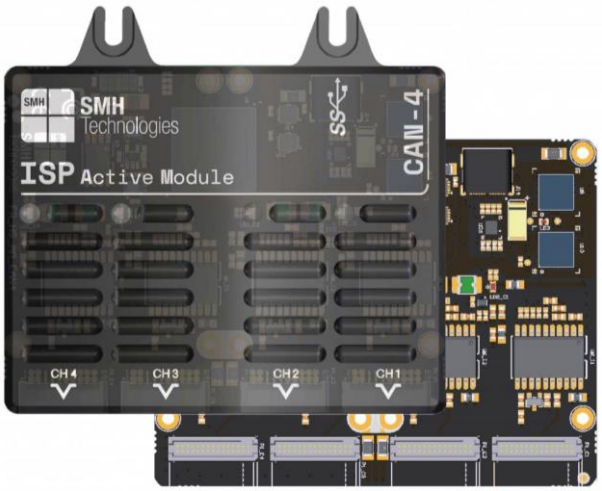
HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

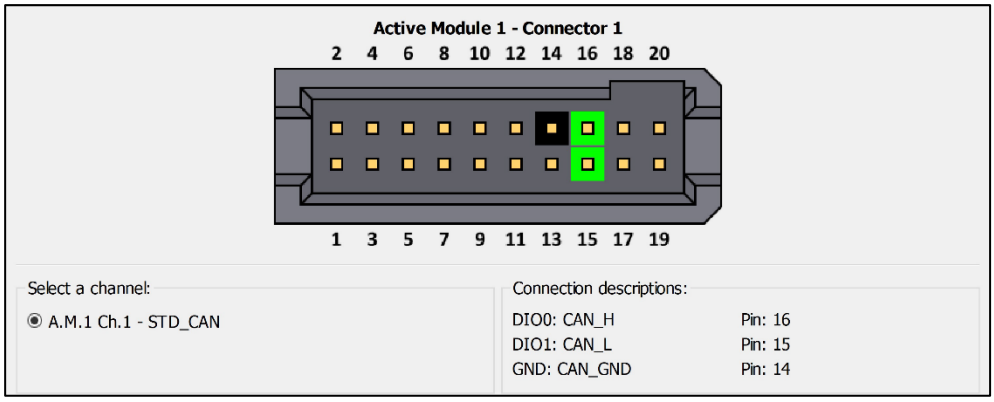
D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

The image on the right shows the CAN-4 Active Module we have developed. This Active Module is specifically designed to allow both CAN and CAN-FD communications, up to 1Mbps for CAN and 12Mbps for CAN-FD which is the maximum frequency defined by the standard. This Active Module is named CAN-4 because it has four parallel independent communications channels. Each one of them is featured by a galvanically isolated transceiver to interface a CAN bus as required by the CAN and CAN-FD protocol specifications.
The CAN-4 module is designed with an additional interface board for the D-SUB 9-pin connectors which could be plugged into the module if needed.



One of the main peculiarities of CAN bus protocol is that two differential lines are used to communicate. In fact, a CAN transceiver is required to transform these special signals into usable information by the programmer. This is the reason why SMH Technologies developed a dedicated Active Module to manage this protocol which is available only for FlashRunner High-Speed.



On the image above you can see the pinout of one connector mounted on the CAN-4 Active Module, and you can see that only three signals are present:

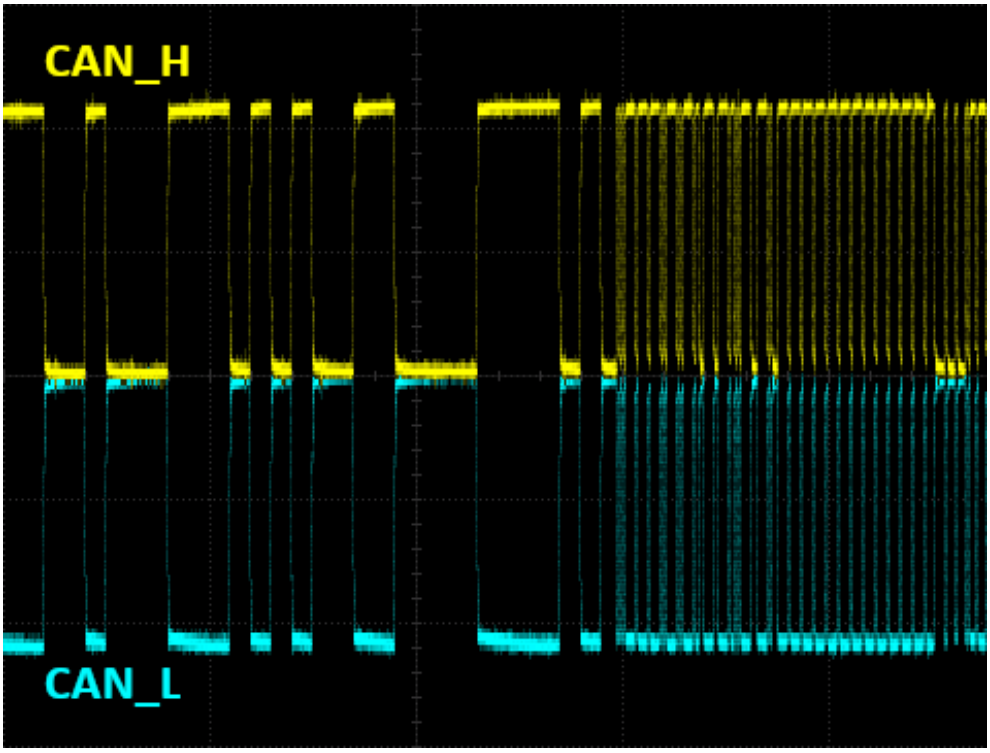
- **CAN_H** and **CAN_L**
  These are the two differential signals which actually carry the data.
- **CAN_GND**
  This is an optional signal which can be useful to have a common ground for the CAN bus nodes. We suggest to connect CAN_GND (when it is possible) to prevent high voltage differences that could damage the nodes connected to the network.
  This ground signal is galvanically isolated from FlashRunner High-Speed.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

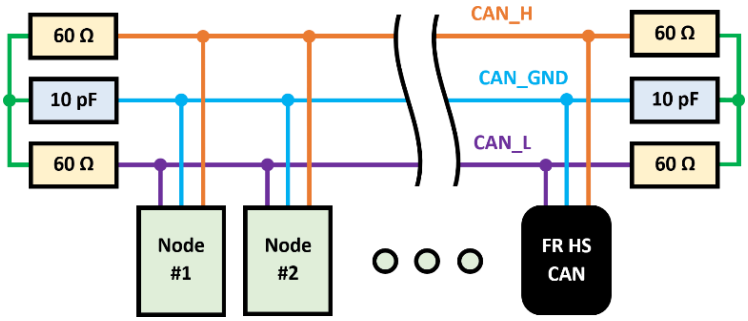
Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com



From the capture above, it is possible to understand the meaning of “differential signals”. In fact, you can see that CAN_H and CAN_L have basically opposite levels and that the CAN bus only allows two valid states: **recessive** (when the voltage difference between CAN_H and CAN_L is about 0V) and **dominant** (when the voltage difference between CAN_H and CAN_L is higher than about 1V).



Another important aspect that customers should consider when realizing the hardware setup is that the CAN bus requires termination load resistors between CAN_H and CAN_L. This does not mean that a resistor must always be placed on the FlashRunner side, it depends on the network in which it is connected. What customers should check is if CAN bus specifications are satisfied. The best solution we suggest is the one shown in the image above where both sides of the network are terminated with 120 Ohm between CAN_H and CAN_L and each termination is split into two 60 Ohm resistors. The split termination improves electromagnetic emissions by adding a low-pass filter for the common-mode noise on the network.

In the typical case when FlashRunner has to communicate with a single target, then it is highly probable that the user needs to place the resistors on both terminations as explained above. The customers also can ask us to mount these resistors on the CAN-4 Active Module when purchasing it.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

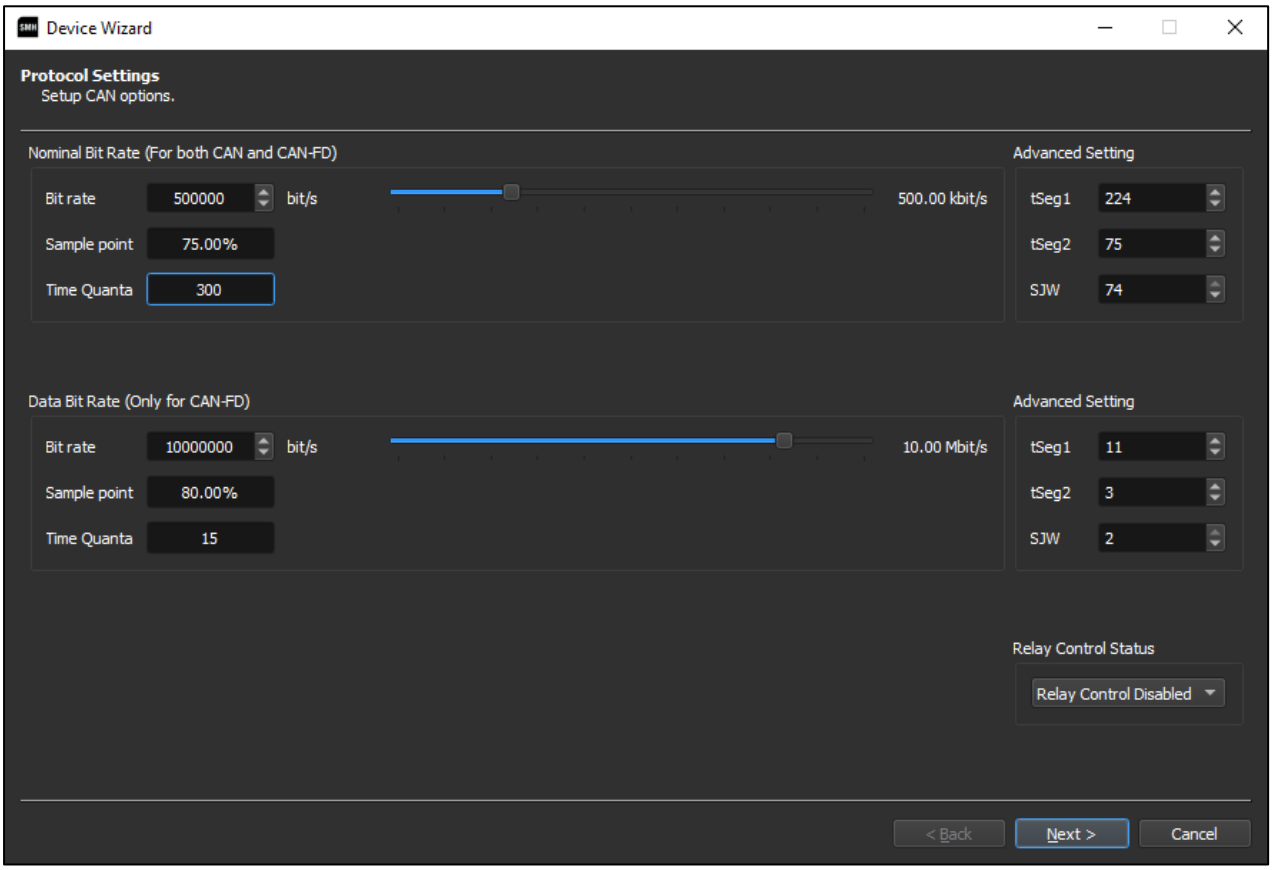
info@smh-tech.com

4. Project Configuration

In the previous chapters we often mentioned CAN and CAN-FD and now it is time to explain the actual differences between them:

- **CAN**
  - One bit rate, up to 1 Mbps.
  - Maximum 8 bytes for each frame.
- **CAN-FD**
  - Two bit rates: the nominal bit rate that corresponds to the standard CAN specifications (up to 1 Mbps) and the data bit rate that is used only when transferring data and can reach up to 12 Mbps.
  - Maximum 64 bytes for each frame.

CAN and CAN-FD bit rates must be the same for all the nodes in the network.
Clarified that, now we can see how to configure a new project using the Workbench. So, just start creating a new project and select “STD_CAN” as device, this will lead you to the following window where you can select the bit rates to use for CAN and CAN-FD. This setting is fundamental to make the communication with other nodes work.



Moreover, there are also some advanced settings that allow you to configure the proper sample point according to your network characteristics. Default values are typically ok but, if you want to be punctilious, you can check that all the nodes share the same settings (or as close as possible).

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

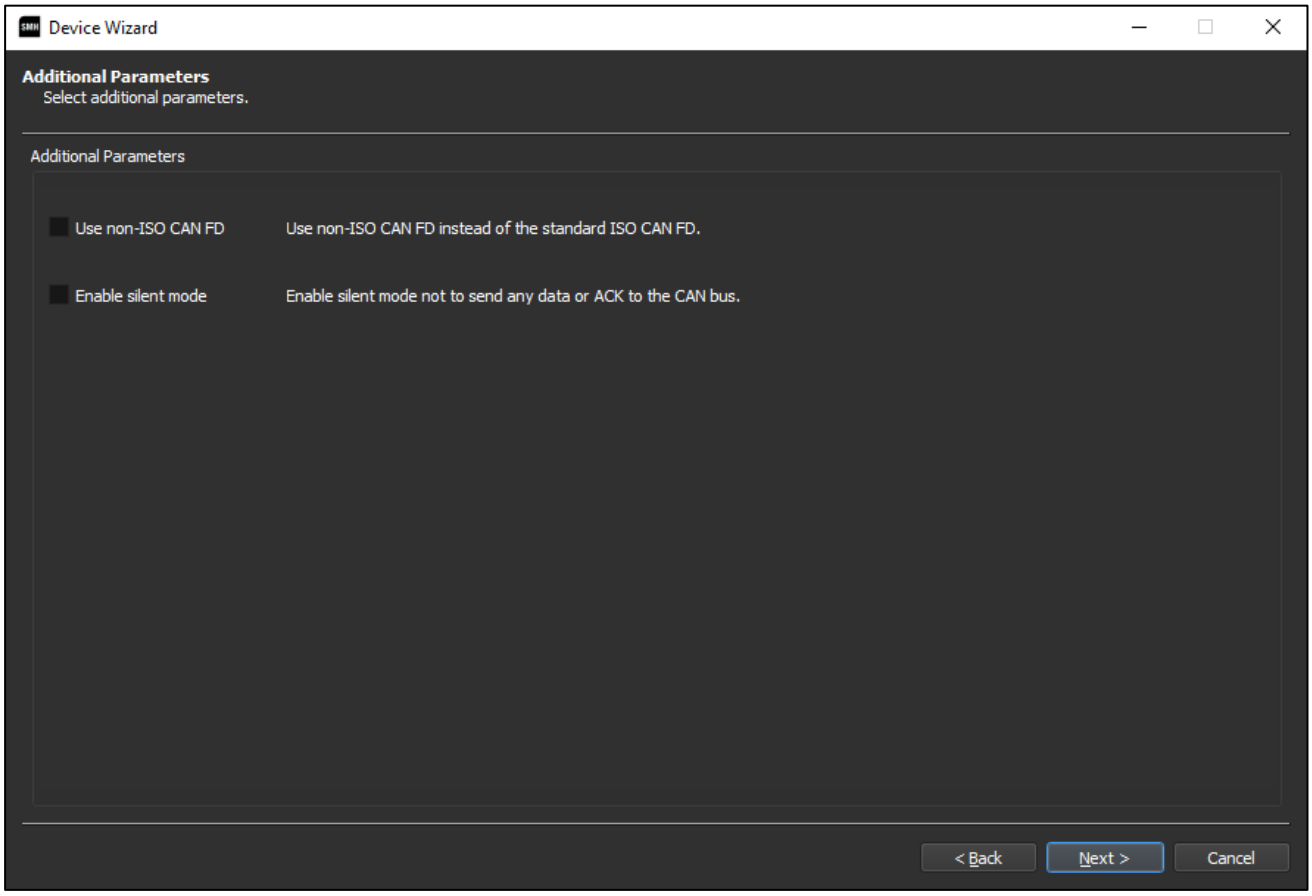
→smh-tech.com
info@smh-tech.com

Warning: after generating the project file, it is not possible to manually edit these parameters from the textual project editor because if you edit them wrongly, then the project execution will fail. In fact, the driver checks if these parameters make sense during the TPSTART command. If you actually need to edit these parameters, please, edit the project using the wizard tool from the Workbench (shortcut: ctrl+alt+e).

From this page, it is also possible to enable or disable the Relay Barrier usage. This can be useful if customers want to totally isolate the target board from the FlashRunner when they are performing the functional test on the same station. Using the Relay Barrier we can connect the FlashRunner to the board when the relays are closed and cut off the programmer from the board when the relays are opened. If you need the Relay Barrier, you need to ask for that when purchasing the CAN-4 Active Module because it is an additional hardware plugin to the module.

On the next page, you will find some additional parameters that you can choose:

- **Use non-ISO CAN-FD**
  This should be enabled in case you need to use the non-ISO CAN-FD instead of the standard ISO CAN-FD. This could be required according to your network configuration.
- **Enable silent mode**
  This should be selected not to send any data or ACK to the CAN bus. This could be useful if you just need to monitor the communications on the bus without interfering.



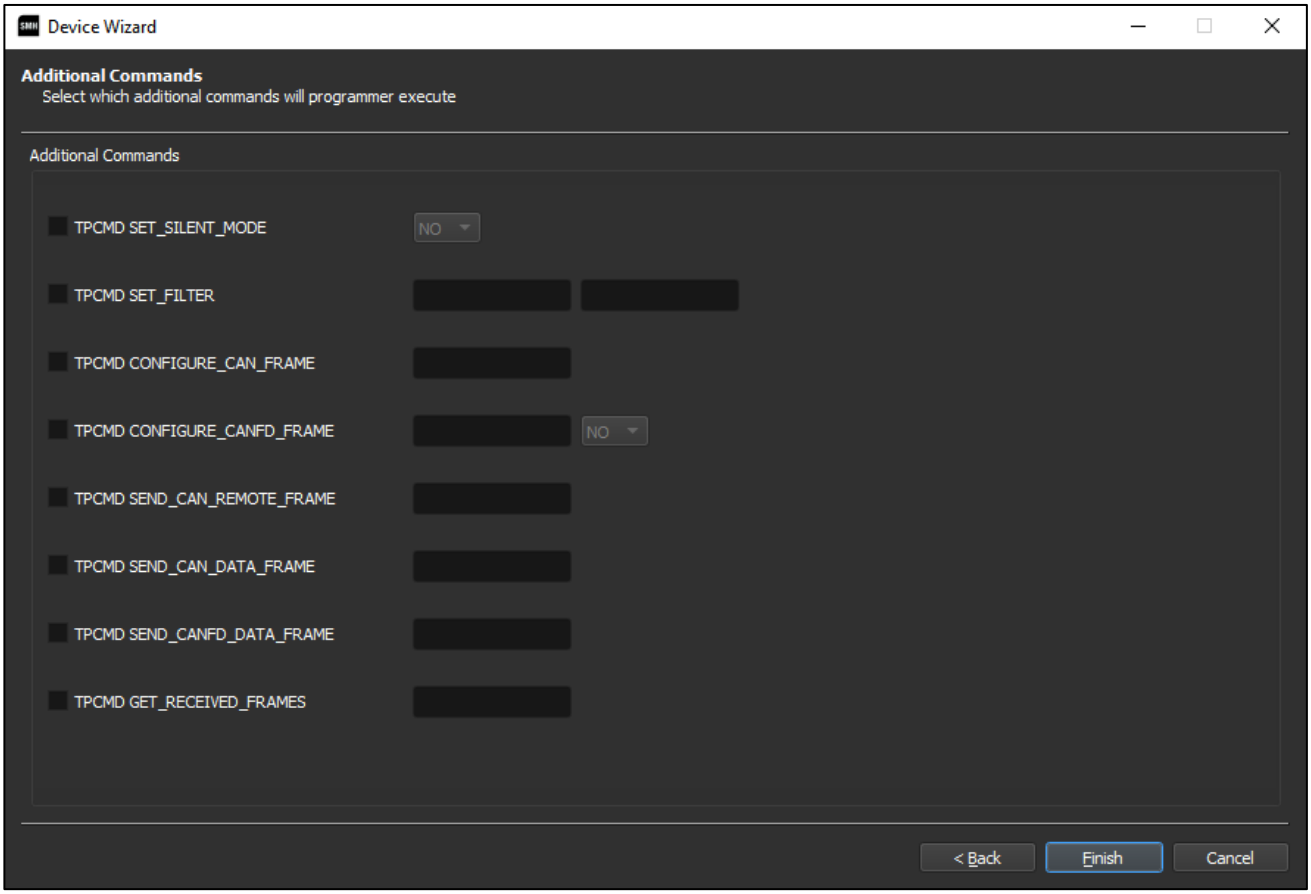
HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

Going forward to the next page, you can find the list of commands that can be used to work with this general-purpose STD_CAN driver, their usage is deeply described in the following chapter.



HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

5. Custom Application

The STD_CAN driver aims to be as generic as possible, so we implemented low-level commands that can be integrated inside a more complex application. For example, the customer can call commands to send and receive CAN frames from his TestStand sequence using the DLL supplied by SMH Technologies. He can develop a high-level application using the low-level commands of this driver and perform any kind of operation: flashing, testing, calibrating, diagnosing, and much more.
Anyway, in case implementing an application in this way results too complex or not enough performing, SMH Technologies can also develop custom drivers to satisfy customers' requests.
Of course, if you need to flash a device with a CAN interface (such as Melexis or Elmos devices), SMH Technologies can you implementing a dedicated algorithm: in this way SMH Technologies will be able to guarantee you that the device works properly according to Silicon Producers' specifications and you do not have to worry about data retention problems.

This is the list of commands that are available for STD_CAN driver:

- **`#TPCMD CONNECT`**
  Activates the CAN transceiver and the FlashRunner channel starts working according to the settings chosen by the user on the previous parameters or commands.

- **`#TPCMD DISCONNECT`**
  Switches off the FlashRunner channel and the CAN transceiver, discarding all data received and ignoring all the upcoming communications.

- **`#TPCMD SET_SILENT_MODE YES`**
  Enables silent mode not to send any data or ACK to the CAN bus. This could be useful if you just need to monitor the communications on the bus without interfering.

- **`#TPCMD SET_SILENT_MODE NO`**
  Disables silent mode, so the FlashRunner can send data and ACK to the CAN bus.

- **`#TPCMD SET_FILTER <Min_ID> <Max_ID>`**
  Filters the received frames according to the range of IDs selected. The first parameter selects the minimum ID value and the second parameter selects the maximum ID. IDs can be expressed in both 11-bit format (standard format) or 29-bit format (extended format) and they must be written as a hexadecimal stream. See some examples below:
  - **`#TPCMD SET_FILTER 2C7 4EF`**
  - **`#TPCMD SET_FILTER 000 7FF`**
  - **`#TPCMD SET_FILTER 0046A9C8 08AEF154`**
  - **`#TPCMD SET_FILTER 00000000 1FFFFFFF`**

- **`#TPCMD SET_FILTER <Fixed_ID>`**
  Filters the received frames selecting only those frames which have the ID expressed by the parameter. The format to use is the same as the previous command.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com
info@smh-tech.com

- **#TPCMD CONFIGURE_CAN_FRAME <ID>**
  Sets up the CAN frame to be sent. Using the first parameter, the user can select which ID to use for the frame. ID can be expressed in both 11-bit format (standard format) or 29-bit format (extended format) and they must be written as a hexadecimal stream. See some examples below:

  - **#TPCMD CONFIGURE_CAN_FRAME 2C7**
  - **#TPCMD CONFIGURE_CAN_FRAME 0046A9C8**

- **#TPCMD CONFIGURE_CANFD_FRAME <ID> <BitRateSwitch>**
  Sets up the CAN-FD frame to be sent. Using the first parameter, the user can select which ID to use for the frame. ID can be expressed in both 11-bit format (standard format) or 29-bit format (extended format) and they must be written as a hexadecimal stream. Using the second parameter, the user can choose to enable or disable the bit rate switch when transmitting data. See some examples below:

  - **#TPCMD CONFIGURE_CANFD_FRAME 2C7 YES**
  - **#TPCMD CONFIGURE_CANFD_FRAME 0046A9C8 NO**

- **#TPCMD SEND_CAN_REMOTE_FRAME <Data_Length>**
  Transmits a CAN remote frame with the value of data length expressed by the parameter (up to 8 bytes). See some examples below:

  - **#TPCMD SEND_CAN_REMOTE_FRAME 4**
  - **#TPCMD SEND_CAN_REMOTE_FRAME 8**

- **#TPCMD SEND_CAN_DATA_FRAME <Data_Stream>**
  Transmits a CAN frame with the data expressed by the parameter (up to 8 bytes). See some examples below:

  - **#TPCMD SEND_CAN_DATA_FRAME 54A938DE**
  - **#TPCMD SEND_CAN_DATA_FRAME DE6BE11AC1A032F0**

- **#TPCMD SEND_CANFD_DATA_FRAME <Data_Stream>**
  Transmits a CAN-FD frame with the data expressed by the parameter. Data stream can be up to 64-byte long and it must be one of the following values: 0, 1, 2, 3, 4, 5, 6, 7, 8, 12, 16, 20, 24, 32, 48, or 64. See some examples below:

  - **#TPCMD SEND_CANFD_DATA_FRAME 54A938DE**
  - **#TPCMD SEND_CANFD_DATA_FRAME DE6BE11AC1A032F0**
  - **#TPCMD SEND_CANFD_DATA_FRAME 54A938DEDE654A938DEBE11AC1A032F0**

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

- **#TPCMD GET_RECEIVED_FRAMES**
  Returns the received frames. Frames are printed out in JSON format to be easily parsed by an application. This works like a FIFO, once the frame is returned, it is removed from FlashRunner memory. See some examples below:

```
#1*TPCMD GET_RECEIVED_FRAMES
01|{
    "FRAME_1" : {
        "isGood" : true,
        "id" : "400",
        "protocol" : "CAN-FD",
        "errorStatusIndicator" : 0,
        "dataLength" : 16,
        "dataStream" : "12345678123456781234567812345678"
    }
}

01|{
    "FRAME_2" : {
        "isGood" : true,
        "id" : "7FE",
        "protocol" : "CAN-FD",
        "errorStatusIndicator" : 0,
        "dataLength" : 20,
        "dataStream" : "00010203000102030001020300010203000102030001020300010203"
    }
}

01|{
    "FRAME_3" : {
        "isGood" : true,
        "id" : "10000000",
        "isRemoteFrame" : false,
        "protocol" : "CAN",
        "dataLength" : 5,
        "dataStream" : "A987654321"
    }
}

01|{
    "FRAME_4" : {
        "isGood" : true,
        "id" : "1FFFFFFF",
        "isRemoteFrame" : false,
        "protocol" : "CAN",
        "dataLength" : 8,
        "dataStream" : "0706050403020100"
    }
}
```

As you can see above, the JSON element can have various fields:

- **FRAME_x**: is the name of the frame given by FlashRunner, where “x” is a number starting from 1 every time the command is called.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

- **isGood**: indicates if the frame is ok (CRC checked successfully) or if it has been corrupted due to the overflow of the FIFO or to other external factors.
- **id**: indicate the ID of the frame. ID can be expressed in both 11-bit format (standard format) or 29-bit format (extended format) according to which format is actually used.
- **isRemoteFrame**: indicates if the frame is a remote frame or not (available only for CAN protocol).
- **protocol**: indicates if the frame has been sent using CAN or CAN-FD protocol.
- **dataLength**: indicates the number of bytes present on the dataStream. In case of remote frames, this indicates the data length requested.
- **dataStream**: contains the data stream expressed in hexadecimal. In case of remote frames, the data stream has no value because remote frames don't have data field.

Additional info:

1. In case no frames have been received before calling this command, the message "No frames received" will be returned.
2. When the FIFO is full of received frames, all the upcoming frames will be discarded, this info will be reported with the message "RX frames lost = x", where "x" is a decimal number. This message can be returned before any frame.
3. To flush the FIFO, you can use the disconnect command followed by the connect command.

- **#TPCMD GET_RECEIVED_FRAMES <Max_Frame_Count>**
  Returns the received frames as the previous command, the only difference is that using the parameter you can specify the maximum number of frames to return. For example, if you set the number to 1, then only the first received frame will be returned.

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

6. Case Study

In this chapter, we are showing an example of an application using the STD_CAN driver and its features.

Let us start with the project file generated from the Workbench using 500 kbps as nominal bit rate and 2 Mbps as data bit rate. The project can be cropped until the #TPSTART command and then the other commands can be sent one by one from the customer application:

```
!ENGINEMASK 0x00000001
#LOADDRIVER libstd_can.so SMH GENERIC STD_CAN
#TCSETDEV VDDMIN 5000
#TCSETDEV VDDMAX 5000
!CRC 0x7751ADA7
#TCSETPAR CANFD_SJW 17
#TCSETPAR CANFD_TSEG1 56
#TCSETPAR CANFD_TSEG2 18
#TCSETPAR CAN_SJW 74
#TCSETPAR CAN_TSEG1 224
#TCSETPAR CAN_TSEG2 75
#TCSETPAR DATA_RATE 2000000
#TCSETPAR FPGA_FREQ 150000000
#TCSETPAR NonISO_CANFD NO
#TCSETPAR PROTCLK 500000
#TCSETPAR SILENT_MODE NO
#TCSETPAR CMODE CAN
#TPSTART
```

To start we can send the connect command which is mandatory to enable the communication:

```
#1*TPCMD CONNECT
```

Let us consider now an example of an application that requires a periodic CAN-FD frame to be sent (using ID = 0x123) and a periodic poll of a response having ID = 0x1E5. So, we can set a filter to only receive frame having ID = 0x1E5:

```
#1*TPCMD SET_FILTER 1E5
```

And we can configure a CAN-FD frame to be sent using ID = 0x123 and enabling bit rate switch:

```
#1*TPCMD CONFIGURE_CANFD_FRAME 123 YES
```


HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com

Now we can create a loop (this is up to your application) that sends periodically 16 bytes of data:

```
#1*TPCMD SEND_CANFD_DATA_FRAME C1A0C1A0C1A0C1A0C1A0C1A0C1A0C1A0
```

And we can also set another loop (this is also up to your application) that get the first frame from the FIFO (you could loop the command if no frames have been received):

```
#1*TPCMD GET_RECEIVED_FRAMES 1
```

You can also flush the FIFO by calling the following command sequence:

```
#1*TPCMD DISCONNECT
#1*TPCMD CONNECT
```

Warning: it is mandatory to call the #TPEND command at the end of the execution to free the memory of the FlashRunner (as it is for any other FlashRunner project).

HQ and Registered Office
Via Giovanni Agnelli 1
33083 Villotta di Chions (PN) Italy
Società Unipersonale

Capitale sociale € 50.000
P.I. 01697470936
C.F. 01697470936
REA PN-97255

D-U-N-S® 33-924-9717
T + 39 0434 421 111
F + 39 0434 639 021

→smh-tech.com

info@smh-tech.com