CANLOADER MANUAL

Rev.	Prepared by		Date	Approved		Date
0.0	RBCS	MM MR	18/04/2008			
0.1	RBCS	MM	12/01/2009			
1.0	RBCS	MR	05/05/2010			

1 Revision history

Rev.	Date	Description
0.0	18/04/2008	Preliminary emission, Basic description of CanLoader messages
0.1	12/01/2009	Added CanLoader communication diagrams
1.0	05/05/2010	Protocol revision (strain board), added description of firmware files, added example logs

2 Summary

3 Introduction

This document describes the CAN protocol used by the CanLoader application.

A description of the messages used by the CanLoader application to download the firmware is presented in sections 4 and 5.

Section 6 describes the syntax of the firmware files (.S Motorola file and .HEX Intel file) parsed by the CanLoader application during the firmware download.

A scheme of the communication protocol between the PC and the boards is described in section 7.

Example logs are provided in section 8.

4 CANLoader messages

According to the standard CAN bus protocol, the structure of the CAN messages used by the CANLoader application is constituted by a header (11bits) and a payload of 8 bytes.

3 bits	4 bits	4 bits	Data[0-7]
Message class	Source	Destination	Payload

Message class is fixed to 111 (0x07) for the CAN loader message type.

Source represents the CAN address of the device which sent the message. Address 0x00 is reserved to identify a message transmitted by the PC. Address 01-14 (0x01-0x0E) represent the IDs of the boards present on the CAN bus.

Destination represents the CAN address of the board to which the command is addressed. According to the standard CAN bus protocol, valid values for the destination field are 01-14 (0x01-0x0E). A special broadcast address 15 (0x0F) is reserved to specify a common request to all the boards present on the CAN bus.

Payload constitutes the content of the message. This field has variable length. Maximum length of the payload is 8 bytes. Data[0] contains the command type. The meaning of Data[1-7] depends on the specific command type (Data[0]).

An explanation of all CAN messages used by the CanLoader application is described in the following sections. The two flags **FIRMWARE** and **BOOTLOADER** will be used to identify if the specific described command is interpreted by the firmware or by the bootloader of the device receiving the CanLoader command.

4.1 CMD_BROADCAST FIRMWARE BOOTLOADER

This message is used to get information (i.e. board type and firmware version) about a specific board.
The CANLoader application also uses this command to get information about the number of boards present on the bus and their firmware version. In this case the command is sent setting the destination field to 0x0F (broadcast message). All the board on the bus will answer to the broadcast message and the CANLoader application will thus collect the data from all the boards.

NOTE: This command behaves differently depending if the board is running the firmware or the bootloader section. See below for additional information.

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
111	0000	Destination	CMD_BROADCAST	NU	NU	NU	NU	NU	NU	NU

RECEIVED FROM THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
111	Source	0000	CMD_BROADCAST	BOARD_TYPE	VERSION	BUILD NUM	NU	NU	NU	NU

Where:

- CMD_BROADCAST = 0xFF;
- BOARD_TYPE =
  - `BOARD_TYPE_DSP 0x00`
  - `BOARD_TYPE_PIC 0x01`
  - `BOARD_TYPE_2DC 0x02`
  - `BOARD_TYPE_4DC 0x03`
  - `BOARD_TYPE_BLL 0x04`
  - `BOARD_TYPE_SKIN 0x05`
  - `BOARD_TYPE_STRAIN 0x06`
  - `BOARD_TYPE_MAIS 0x07`
  - `BOARD_UNKNOWN 0xFF`

FIRMWARE

- VERSION = version of the firmware.
  Different firmware versions implement different controllers/behaviors.
- BUILD NUM = build number (revision) of the firmware.

BOOTLOADER

- VERSION = version of the bootloader.
- BUILD NUM = bootloader revision.

4.2 CMD_BOARD FIRMWARE BOOTLOADER

This command has different meaning depending if its received by the firmware or by the bootloader.

In the firmware, this command is used to make the device jump to the bootloader section (the controller execution is effectively terminated). The bootloader section replies to this command appropriately by sending a 0x00, 0x01 back to the sender.

After the jump, the execution of the bootloader section will last for 5 seconds. If no command is received during these five seconds, then the device will terminate the execution of the bootloader section and will jump again to the firmware code. On the contrary, if a new CMD_BOARD command is received by the bootloader, the timer is stopped and the device will remain in the bootloader section until a CMD_END is received. In this latter case, an EEPROM_FLAG is used to indicate if the eeprom of the board has to be rewritten during the firmware download.

If a multiple download is required (i.e: the same firmware flashed on multiple boards) the CANLoader application sends a CMD_BOARD command to all the specified boards in order to execute the jump to the bootloader section. The following CMD_ADDRESS, CMD_DATA etc. will be then transmitted in broadcast. In this way, all the devices executing the bootloader will simultaneously receive the transmitted firmware.

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
111	0000	Destination	CMD_BOARD	EEPROM_ FLAG	NU	NU	NU	NU	NU	NU

RECEIVED FROM THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
111	Source	0000	CMD_BOARD	0x01	NU	NU	NU	NU	NU	NU

Where:

- CMD_BOARD = 0x00;
- EEPROM_FLAG = is used to indicate if the eeprom of device will be rewritten during the firmware download. It can be 0x00 (do not touch EEPROM) or 0x01 (write EEPROM). This byte is parsed by the bootloader only, while it is ignored by the firmware when the command is used to make the jump to the bootloader section.

4.3 CMD_ADDRESS **BOOTLOADER**

This command is used to specify the address, type and size of the next firmware data packet.
It must be immediately followed by a CMD_DATA message.
This command is typically sent in broadcast (destination = 0x0F) in order to download the firmware to multiple boards at the same time (all the boards executing the bootloader will received the CMD_ADDRESS message)

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
111	0000	Destin	CMD_ADDRESS	DATA_LEN	ADDR_L	ADDR_H	DATA_TYPE	UADDR_L	UADDR_H	NU

RECEIVED FROM THE BOARD:

NO MESSAGE

Where:

- CMD_ADDRESS = 0x01;
- DATA_LEN = the length of the data block
- ADDR_L = the lower part of the memory address
- ADDR_H = the higher part of the memory address
- DATA_TYPE = block type (program memory 0 or data memory 1).
- UADDR_L = the lower part of the upper 16 bits of the 32 address (only if it is a hex/Intel).
- UADDR_H = the higher part of the upper 16 bits of the 32 address (only if it is a hex/Intel).

4.4 CMD_DATA **BOOTLOADER**

A data block containing part of the program (or memory) data that will be flashed on the target device. A CMD_DATA message contains 6 bytes of payload. For each CMD_ADDRESS, several CMD_DATA messages are consecutively transmitted, depending on the length of the data block (specified in the CM_ADDRESS message). An ACK message from the board is sent at the end of the transmitted data block. This command is typically sent in broadcast (destination = 0x0F) in order to download the firmware to multiple boards at the same time (all the boards executing the bootloader will received the CMD_DATA message)

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
111	0000	Destin	CMD_DATA	DATA	DATA	DATA	DATA	DATA	DATA	NU

RECEIVED FROM THE BOARD (only at the end of the data block):

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
111	Source	0000	CMD_DATA	NU	NU	NU	NU	NU	NU	NU

Where:

- CMD_DATA = 0x03;
- DATA = the program or memory data to be written in the flash of the device

---

Example of block transmission:

070F 01 10 50 02	*CMD_ADDRESS (len = 0x10 = 16 bytes)*
070F 03 16 00 37 00 62 00 070F 03 40 00 E0 80 48 00 070F 03 91 01 88 00	CMD_DATA (6 bytes) CMD_DATA (6 bytes) CMD_DATA (4 bytes)
07E0 03 01 07A0 03 01	ACK from board 0xE (13) *ACK from board 0xA (10)*

4.5 CMD_START **BOOTLOADER**

This command is sent as last command when all the firmware lines have been successfully transmitted to the board. It indicates to the board the firmware transmission is completed, and confirms that it can be safely copied from the internal memory to the flash area. If this command is not received (this can happen if the firmware transmission is canceled/interrupted etc) all the received firmware data will be discharged. In this way the incomplete firmware will be not saved in the flash memory.

The message has the following structure:

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
111	0000	Destin	CMD_START	0x00	0x00	0x00	0x00	NU	NU	NU

RECEIVED FROM THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
111	Source	0000	CMD_START	0x02	0x01	NU	NU	NU	NU	NU

Where:

- CMD_START = 0x02;

4.6 CMD_END **BOOTLOADER**

This command is sent to the board when the download of the firmware is finished.
The command terminates the bootloader message parser and exits the bootloader section, starting the execution of the firmware.

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
111	0000	Destin	CMD_END	NU	NU	NU	NU	NU	NU	NU

RECEIVED FROM THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
111	Source	0000	CMD_END	0x01	NU	NU	NU	NU	NU	NU

Where:

- CMD_END = 0x04

5 Additional messages

There following messages are used by the CANLoader application in order to implement additional functionalities such as changing the CAN address of the board, setting/retrieving additional text information about the board etc.

IMPORTANT NOTE: Since these messages are not involved in the process of the firmware download, they do not belong to the CANLOADER message class (0x07). They are thus considered as standard polling messages (class 0x00 for CONTROL_BOARDS, 0x02 for STRAIN_BOARDS).

5.1 CAN_GET_ADDITIONAL_INFO FIRMWARE

This message is sent by the CAN loader to get the additional information of the board (a user defined text string of 32 characters typically the position of the board in the robot, i.e: LEFT_ARM)

The answer of to board to the CAN_GET_ADDITIONAL_INFO command is splitted in 8 messages, each of them containing 4 characters and a counter (in order to the reconstruct the complete string, reordering the eight messages).

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
000	0000	Destination	CAN_GET_ADD_INFO	NU	NU	NU	NU	NU	NU	NU

RECEIVED FROM THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
000	Source	0000	CAN_GET_ADD_INFO	COUNTER	ADD_INFO	ADD_INFO	ADD_INFO	ADD_INFO	NU	NU

Where:

- CAN_GET_ADDITIONAL_INFO = 12 (0x0C);
- COUNTER is the number of the message, from 0 to 7. It is a progressive number used to specify the order of the additional info messages.
- ADD_INFO = four chars of the string to be sent.

5.2 CAN_SET_ADDITIONAL_INFO **FIRMWARE**

This message type is sent by the CAN loader to set the additional information about the board, for example: position of the board in the robot, etc.
The additional info transmitted to the board is divided into 8 messages, each of them including 4 of the 32 characters of the text and a counter identifier, used by the board to the reconstruct the text (reordering the received messages).

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
000	0000	Destination	CAN_SET_ADD_INFO	COUNTER	ADD_INFO	ADD_INFO	ADD_INFO	ADD_INFO	NU	NU

RECEIVED FROM THE BOARD:

NO MESSAGE

Where:

- CAN_SET_ADDITIONAL_INFO = 13 (0x0D);
- COUNTER is the number of the message, from 0 to 7. It is a progressive number used to specify the order of the additional info messages.
- ADD_INFO = four chars of the string to be sent.

5.3 CAN_GET_BOARD_ID **FIRMWARE**

This message is sent to the board for obtain its current CAN address. The command is typically used together to CAN_SET_BOARD_ID during the process of changing the CAN address of the board.
The message has the following structure:

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
000	0000	Destination	CAN_GET_BOARD_ID	NU	NU	NU	NU	NU	NU	NU

RECEIVED FROM THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
000	Source	0000	CAN_GET_BOARD_ID	BOARD_ID	NU	NU	NU	NU	NU	NU

Where:

- CAN_GET_BOARD_ID = 51 (0x33);
- BOARD_ID= The CAN ID of the board. The range is from 1 to 15.

5.4 CAN_SET_BOARD_ID **FIRMWARE**

This message is used to change the CAN address of the board.
The message has the following structure:

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
000	Source	Destination	CAN_SET_BOARD_ID	NEW_ID	NU	NU	NU	NU	NU	NU

RECEIVED FROM THE BOARD:

NO MESSAGE

Where:

- CAN_SET_BOARD_ID = 50 (0x32);
- NEW_ID = the new ID of the board. According to the CAN bus protocol specifications, accepted values are: 1-15.

5.5 CAN_GET_SERIAL_NUMBER **STRAIN ONLY** **FIRMWARE**

This message is used to retrieve the serial number saved in the eeprom of a strain board.
The serial number is represented by a string of 7 text characters (i.e: **`S/N:006`**).
The message is currently implemented only for strain board, (CAN message class is 0x02)
The message has the following structure:

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
010	Source	Destination	CAN_GET_SERIAL_NUM	SERIAL_N	SERIAL_N	SERIAL_N	SERIAL_N	SERIAL_N	SERIAL_N	SERIAL_N

RECEIVED FROM THE BOARD:

NO MESSAGE

Where:

- CAN_GET_SERIAL_NUMBER ID = 26 (0x1A);
- SERIAL_N = the serial number of the board (7 chars)

5.6 CAN_SET_SERIAL_NUMBER **STRAIN ONLY** **FIRMWARE**

This message is used to set the serial number of a strain board. The serial number must be subsequently saved in the eeprom sending the specific strain command CAN_CMD_SAVE2EE.
The serial number is represented by a string of 7 text characters (i.e: **`S/N:006`**).
Since the message is currently implemented only for strain board, the CAN message class is 0x02 (010)
The message has the following structure:

SENT TO THE BOARD:

3 bits	4 bits	4 bits	Data[0]	Data[1]	Data[2]	Data[3]	Data[4]	Data[5]	Data[6]	Data[7]
200	Source	Destination	CAN_SET_SERIAL_NUM	SERIAL_N	SERIAL_N	SERIAL_N	SERIAL_N	SERIAL_N	SERIAL_N	SERIAL_N

RECEIVED FROM THE BOARD:

NO MESSAGE

Where:

- CAN_SET_SERIAL_NUMBER = 25 (0x19);
- SERIAL_N = the serial number of the board (7 chars)

6 Description of the firmware files

In this section is shown how the the firmware text files are parsed by the CANLoader application. Two different type of files exist (.S / .HEX extension). The Motorola protocol is used by the DSP control boards (MC4, BLL) mounting the DSP568007. The Intel protocol is used by the sensor boards (STRAIN, MAIS) mounting the DSPIC 30f4013.

In the following examples, different colors are used to highlight the parts of the line and identify its meaning, according to the used protocol.

6.1 Motorola protocol (S-REC format)

red: record type. Can be S0 (start), S3 (data record), S7 (end) green: record length (remaining words in the line, from memory type to checksum included) cyan: memory type. Can be 0000 program memory or 0001 data memory blue: memory address	Used by: CMD_ADDRESS
orange: program data. Data messages are splitted in packets of 6 bytes each. (red) grey: memory data	Used by: CMD_DATA
Violet: checksum: the least significant byte of the ones' complement of the sum of the values of all fields. e.g. 05+00+00+00+42 => B8	

Extract from 2BLL.15.out.S

```
S0110000000050524F4752414D2644415441 96
S3290000000484E91E5D84E91E5D84E91E5D84E91E5D84E91E5D84E91E5D84E91E5D84E91E5D84E91E5DAA
S3290000001684E91E5D84E91E5DC8E9485AC8E92953C8E97753C8E9C053C8E9715484E91E5D84E91E5DEB
S3290000002884E91E5D84E91E5D84E91E5D84E91E5D84E91E5D84E91E5D84E91E5D84E91E5D84E91E5D86
S3290000003A84E91E5DC8E9354084E91E5D84E91E5D84E91E5D84E91E5D84E91E5D84E91E5D84E91E5D36
S3290000004CC8E94B46C8E9E846C8E9854784E91E5DC8E97444C8E91145C8E9AE45C8E9DB0084E91E5D7C
S3290000005E84E91E5D84E91E5D84E91E5D84E91E5DC8E9214DC8E9594DC8E9EC4CC8E96C4AC8E9134A04
S3250000007084E91E5D84E91E5D84E91E5D84E91E5DC8E99B43C8E9844384E91E5D84E91E5DF3
S32900000080C8E9840000000500D88300017FC080849B8F9B8F00C97B9954D93013D0879A0A40E03F98EA
S3290000009 2908700C914D9F486A0138100F486A11313F0C08700401164008E03A3F48AA2137A40F48601
S329000000A4A013820054F0A21354D0A21354F999020BDE1FD9DD830001DD810002C8E94361C8E95561E5
S329000000B600C1D2879A02D3879A02C8E9C10741E0D8EDF4861E112000F4861F110040F4861B110000BC
S329000000C8F48618113F9CF4801E11001EEF4821E110010F4861D110000F4801E1100E0F4821E1100202A
S329000000DAD8EDC8E98E5CCC87FFFFD8812000D8830001D881100038BD0BDE1FDD39BD0BDE1FDD3ABD61
...
S329000060E6CBF8F9FFF89ED8EDFBB4C68700FF307CCBF8F9FFF89ED8EDFEB4CBF0FDFF12A9FBB4C6879D
S329000060F880FF307CF48298024000CBF8F9FFF89ED8ED078E02A4068E6DA41FF4CBF0FFFFC68B704040
S3290000610ACBF8F9FFF89ED8ED00C0FC97CBD1FBFF0783A37E0388FE96CBD0FDFFC8E92161806CF89361
S3290000611C35763576FCB4CBF0FBFF08C1068908833C677C7D1183B776FF750DA3C05FFF0005A3C6832B
S3290000612E0080CA870000D8ED0246307401A30146D8ED068E02A2088E7AA301C3008E78A5815730743F
S3290000614006 8E7CA4D8EDC0870100008E0DA3C1875B02018E09A3D3870020D1874000 40E003F001D060
S3230000615213647CA2D8EDC187FF06018E07A300C0D1879B0240E001D013647DA2D8ED48
S3290020200001004500520052003A0020006100780025006400200 05F0068006900670068002000 6300 4E
S3290020201275007200720020004400490053002000500057004D0000000100010001001000240044009C
S3290020202449004100530042004C004500200042005500530020004F0046004600000041004 2005300E7
S3290020203620006500720072006F00720020002500640000004E006F0020006300700 06C0020007000C1
...
S3290020220A3C0020003F0000003F0000003C0020000F0008003C0008003300020033002000 0F00020060
S3290020221C3F0000000000050003000100060004000200000000000300060002000500010004000000 0F
S3290020222EFFFFD011E011F011010028006E0075006C006C0029000000581B0000009435775 81B000062
S3290020224 0581B00 00581B0000581B00001F85EB3E1F85EB3E1F85EB3E1F85EB3E300D0000300D0000 4D
S3170020225 2300D00 00300D000001000100000022 5D0000 59
S7050000 0042B8
```

Here starts the program memory area.
Address ranges:
from: 0000 0000
To: 0000 6152

Here starts the data memory area.
Address ranges:
from: 0020 2000
To: 0020 2252

6.2 Intel protocol (.Hex file)

green: data length (hex lenght of the program data in words) red: record type. Can be 00 (standard data record), 01 (end of file), 04 (extended linear address record). Cyan: upper 16 bits of a 32 bit address record (only in 04 records, combined with a 00 record) blue: lower 16 bits of a 32 bit address record (00 record)	Used by: CMD_ADDRESS
orange: program data. Data messages are splitted in packets of 6 bytes each. (red)	Used by: CMD_DATA
Violet: checksum: the least significant byte of the two's complement of the sum of the values of all fields. e.g. 02+00+00+cf+bb+20+00+80+f0+20+00+00+01+88+00+00+00+00 => 2b	

Extract from strain.hex:

```
:02000004 0000fa
:08000000 00010400 00000000f3
:02000004 0000fa
:10020000 cfbb2000 80f02000 00188000 00000002b
:10021000 05000700 0c000700 ea0b0200 00000000c8
:100220000040da000000fe004440a900c0002000a9
:100230000000e0000300320000002000a001880060
:100240004440a8000000060040fe210001002000fc
:1002500016003700620040000e080480091018800ed
...
:103f600006003700ca62a800ca82a900caa2a80037
:103f700002003700100200099fe07000080fa00c0
:103f800000000600f03fb1000180b10006003500de
:103f9000ee03090000000000403fb1000180b100c5
:103fa000fbff3d001000b000203fb00002003500d4
:0c3fb000008009000000000000060076
...
:02000004 00fffb
:10fa4c00 20000000e3000000dd000000d6680000aa
:10fa5c00e3000000 87d40000 43030000d5fd0000 44
:10fa6c008f7200004303000014c900000000000066
:10fa7c00000000000000000000000000000000007a
:10fa8c00000000000000000000000000000000006a
:10fa9c00000000000000000000000000000000005a
:08faac0000000000000000052
:00000001FF
```

A 04 record is used to specify the upper 16 bits of the address. The following 00 record contains the lower 16 bits and the data. The resulting address are here reported:

```
0000 0200
0000 0210
0000 0220
...
...
0000 3f90
0000 3fa0
0000 3fb0
...
...
00ff fa4c
00ff fa5c
00ff fa6c
...
```

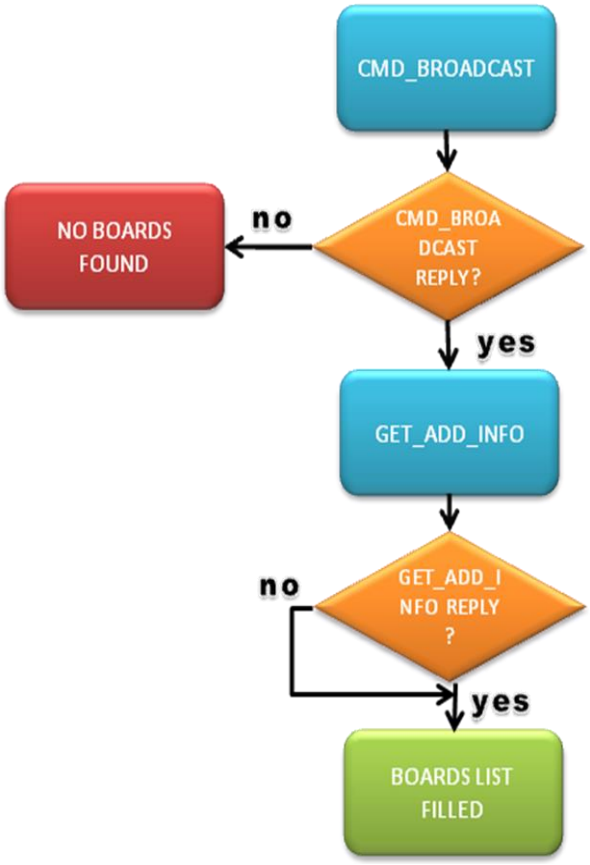
7 CANLoader communication scheme

The schemes presented in this section describe the communication protocol between the CANLoader application and the boards.

HOW TO READ THE SCHEMA (color legend)

7.1 Connection of the CANLoader

The scheme below describes the algorithm used by the CanLoader application to obtain the list of the boards present on the CAN bus.



Description:

1. The application sends a CMD_BROADCAST with destination 0x0F (all boards)
2. Each board present on the bus answer to CANLoader specifying its type, firmware version etc.
3. The application collects the messages and populates a list of the boards present on the bus.

4. For each board in the list, a specific GET_ADDITIONAL_INFO commands is sent
5. The board answers specifying its additional info

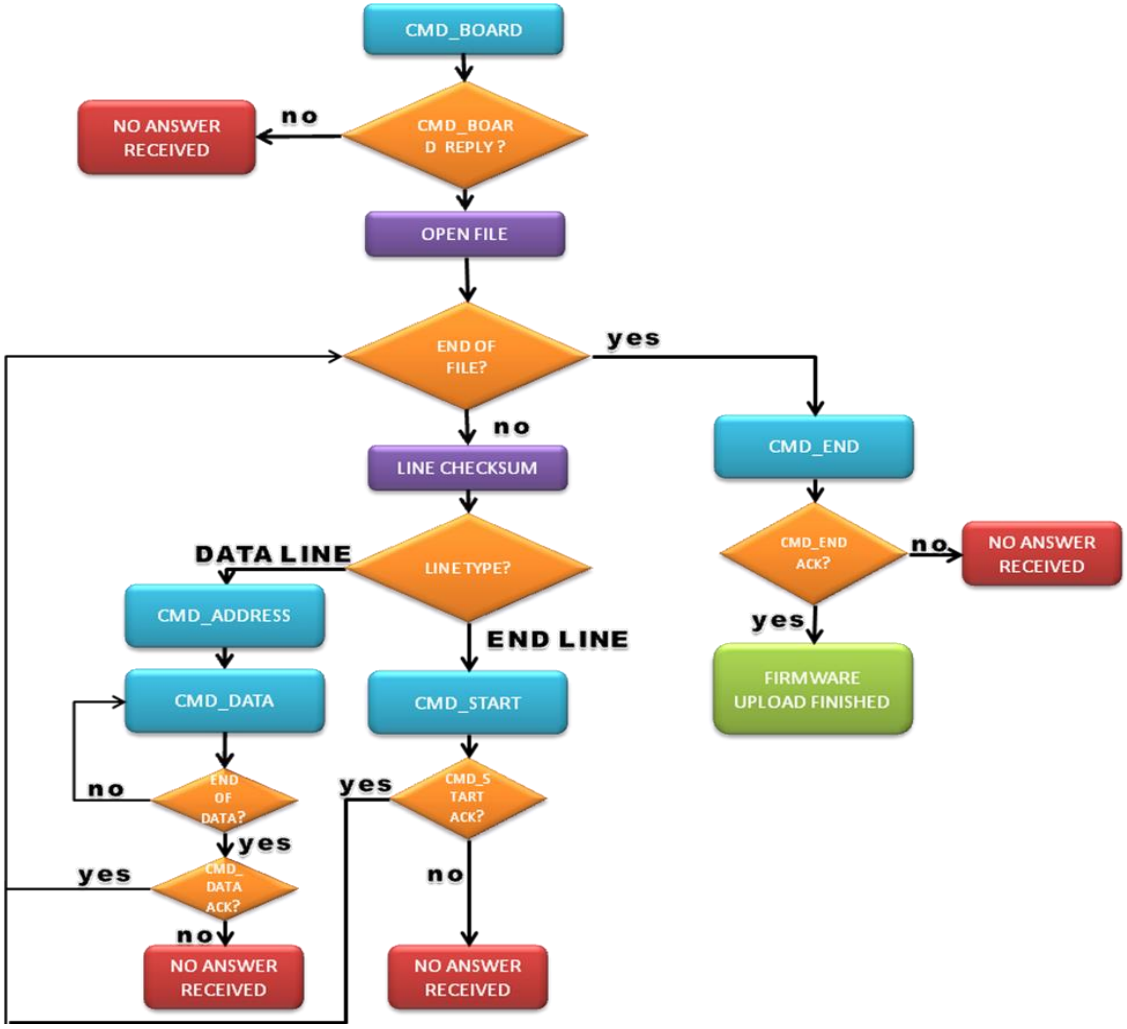
   TODO: (currently implemented only for STRAIN BOARDS)
6. For each board in the list, a specific GET_SERIAL_NUMBER commands is sent
7. The board answers specifying its serial number

Example:

See section 8.1 for an example log.

7.2 Firmware download

The scheme below describes the algorithm used by the CanLoader application to download the new firmware in a target device(s).



Description:

1. The application sends a CMD_BOARD to each board that will receive the firmware to make it jump to the bootloader section.
2. Again, a CMD_BOARD command is sent to all the boards running the bootloader section, in order to stop their timer and make them remain in the bootloader until the download is finished.
3. The firmware is downloaded to the boards. A CMD_ADDRESS is sent (destination address = broadcast 0x0F), followed by one or more CMD_DATA (destination address= broadcast 0x0F) containing the firmware. Only the firmware running the bootloader will receive the firmware sent in broadcast. An CMD_DATA_ACK is sent by each board when the data block is completely received
4. After the complete download of the firmware a CMD_START data is sent (destination address = broadcast 0x0F) in order to inform the boards that the download is complete and the received data can be copied in the flash memory
5. A CMD_END command is sent (destination address = broadcast 0x0F) to make the board terminate the execution of the bootloader section.

Example:

See section 8.2 for an example log.

8 Logs

For better clarity, in this section are presented different logs representing the communication between the PC and the devices present on the CAN bus during different operation of the CANLoader application.

The messages highlighted in **red** represent the commands sent by the application.

The messages highlighted in **orange** represent the messages CMD_ADDRESS / CMD_DATA sent by the CANLoader application during the firmware transmission.

The messages in black represent the answer from the boards.

8.1 CanLoader connection

In this example, an MC4 board (address 0x03) and a STAIN board (address 0x0E) are present on the can bus.

Num	Rel	IdHex	Len	d1	d2	d3	d4	d5	d6	d7	d8	Text
1	**0.0**	**070F**	**1**	**FF**	**(CMD_BROADCAST)**							**ÿ**
2	0.127	07E0	5	FF	06	02	03	06				ÿ____
3	0.626	0730	5	FF	03	01	11	17				ÿ____
4	**1278.002**	**070E**	**1**	**0C**	**(GET_ADDITIONAL_INFO)**							**_**
5	0.143	07E0	6	0C	00	00	01	02	03			_____
6	0.113	07E0	6	0C	01	04	05	06	07			_____
7	0.113	07E0	6	0C	02	08	09	0A	0B			_____
8	0.177	07E0	6	0C	03	0C	0D	0E	0F			_____
9	0.047	07E0	6	0C	04	10	11	12	13			_____
10	0.123	07E0	6	0C	05	14	15	16	17			_____
11	0.098	07E0	6	0C	06	18	19	1A	1B			_____
12	0.111	07E0	6	0C	07	1C	1D	1E	1F			_____
13	**988.320**	**0703**	**1**	**0C**	**(GET_ADDITIONAL_INFO)**							**_**
14	0.659	0730	6	0C	00	62	6F	61	72			__boar
15	0.100	0730	6	0C	01	64	20	31	00			__d_1_
16	0.105	0730	6	0C	03	00	00	00	00			_____
17	0.106	0730	6	0C	04	00	00	00	00			_____
18	0.104	0730	6	0C	05	00	00	00	00			_____
19	0.137	0730	6	0C	06	00	00	00	00			_____
20	0.073	0730	6	0C	07	00	00	00	00			_____
21	0.104	0730	6	0C	02	00	00	00	00			_____
22	**988.602**	**020E**	**1**	**1A**	**(GET_SERIAL_NUMBER)**							**_**
23	0.158	02E0	8	1A	53	4E	30	32	33	00	00	_SN023__

8.2 Firmware download (Strain board)

In this example, an STRAIN board (address 0x0D) is flahed by the CANLoader application.

Num	Rel	IdHex	Len	d1	d2	d3	d4	d5	d6	d7	d8	Text
1	0.0	070D	2	00	00	(CMD_BOARD, JUMP TO BOOTLOADER)						__
2	1500.1	070D	2	00	00	(CMD_BOARD, START DOWNLOAD)						__
3	0.086	07D0	2	00	01							__
4	1485.0	070F	7	01	08	00	00	00	00	00		_____
5	10.926	070F	7	03	00	01	04	00	00	00		_____
6	5.951	070F	3	03	00	00						___
7	665.209	07D0	2	03	01							__
8	1.688	070F	7	01	10	00	02	00	00	00		_____
9	11.153	070F	7	03	CF	BB	20	00	80	F0		_Ï»__€ð
10	5.998	070F	7	03	20	00	00	01	88	00		_____ˆ_
11	5.972	070F	5	03	00	00	00	00				_____
12	0.099	07D0	2	03	01							__
13	5.952	070F	7	01	10	10	02	00	00	00		_____
14	10.971	070F	7	03	05	00	07	00	0C	00		_____
15	5.994	070F	7	03	07	00	EA	0B	02	00		___ê___
16	5.989	070F	5	03	00	00	00	00				_____
17	0.101	07D0	2	03	01							__
18	5.983	070F	7	01	10	20	02	00	00	00		_____
19	10.946	070F	7	03	00	40	DA	00	00	00		__@Ú___
20	5.995	070F	7	03	FE	00	44	40	A9	00		_þ_D@©_
21	5.986	070F	5	03	C0	00	20	00				_À___
22	0.098	07D0	2	03	01							__
...												
5522	5.983	070F	7	01	10	8C	FA	00	FF	00		__Œú_ÿ_
5523	11.953	070F	7	03	00	00	00	00	00	00		_____
5524	5.985	070F	7	03	00	00	00	00	00	00		_____
5525	5.990	070F	5	03	00	00	00	00				_____
5526	0.097	07D0	2	03	01							__
5527	6.964	070F	7	01	10	9C	FA	00	FF	00		__œú_ÿ_
5528	11.959	070F	7	03	00	00	00	00	00	00		_____
5529	7.001	070F	7	03	00	00	00	00	00	00		_____
5530	5.970	070F	5	03	00	00	00	00				_____
5531	0.097	07D0	2	03	01							__
5532	6.019	070F	7	01	08	AC	FA	00	FF	00		__¬ú_ÿ_
5533	11.916	070F	7	03	00	00	00	00	00	00		_____
5534	5.959	070F	3	03	00	00						___
5535	0.089	07D0	2	03	01							__
5536	5.969	070F	5	02	00	00	00	00 (CMD_START, SAVE FLASH)				_____
5537	0.085	07D0	2	02	01							__
5538	984.645	070F	1	04		(CMD_END, EXIT BOOTLOADER)						_
5539	0.084	07D0	2	04	01							__
5540	806.597	02D0	6	00	00	00	01	00	00			_____

8.3 Firmware download (MC4 board)

In this example, two MC4 boards (address 0x04, 0x06) are simultaneously flahed by the CANLoader application.

Num	Rel	IdHex	Len	d1	d2	d3	d4	d5	d6	d7	d8	Text
1	0.0	0704	1	00		(CMD_BOARD, JUMP TO BOOTLOADER)						_
2	0.140	0740	2	00	01							__
3	250.89	0704	1	00		(CMD_BOARD, START DOWNLOAD)						_
4	0.074	0740	2	00	01							__
5	1478.87	0706	1	00		(CMD_BOARD, JUMP TO BOOTLOADER)						_
6	0.142	0760	2	00	01							__
7	251.22	0706	1	00		(CMD_BOARD, START DOWNLOAD)						_
8	0.078	0760	2	00	01							__
9	1483.6	070F	5	01	24	04	00	00				_$___
10	1.114	070F	7	03	84	E9	BC	56	84	E9		_„é¼V„é
11	1.893	070F	7	03	BC	56	84	E9	BC	56		_¼V„é¼V
12	1.994	070F	7	03	84	E9	BC	56	84	E9		_„é¼V„é
13	2.006	070F	7	03	BC	56	84	E9	BC	56		_¼V„é¼V
14	1.989	070F	7	03	84	E9	BC	56	84	E9		_„é¼V„é
15	2.011	070F	7	03	BC	56	84	E9	BC	56		_¼V„é¼V
16	0.133	0740	2	03	01							__
17	0.068	0760	2	03	01							__
18	1.914	070F	5	01	24	16	00	00				_$___
19	1.954	070F	7	03	84	E9	BC	56	84	E9		_„é¼V„é
20	1.001	070F	7	03	BC	56	C8	E9	33	4E		_¼VÈé3N
21	1.001	070F	7	03	C8	E9	FF	46	C8	E9		_ÈéÿFÈé
22	1.988	070F	7	03	4D	47	C8	E9	96	47		_MGÈé–G
23	1.009	070F	7	03	C8	E9	47	48	84	E9		_ÈéGH„é
24	2.002	070F	7	03	BC	56	84	E9	BC	56		_¼V„é¼V
25	0.100	0740	2	03	01							__
26	0.070	0760	2	03	01							__
...												
10640	1.913	070F	5	01	24	76	22	01				_$v"_
10641	1.986	070F	7	03	25	00	64	00	00	00		_%_d___
10642	2.012	070F	7	03	40	06	00	00	00	6A		_@____j
10643	2.001	070F	7	03	18	00	00	6A	18	00		____j__
10644	1.991	070F	7	03	00	6A	18	00	00	6A		__j___j
10645	1.002	070F	7	03	18	00	82	E2	47	3D		____,âG=
10646	1.992	070F	7	03	82	E2	47	3D	82	E2		_,âG=,â
10647	0.100	0740	2	03	01							__
10648	0.068	0760	2	03	01							__
10649	1.990	070F	5	01	0C	88	22	01				__ˆ"_
10650	0.978	070F	7	03	47	3D	82	E2	47	3D		_G=,âG=
10651	0.972	070F	7	03	00	00	C0	56	00	00		___ÀV__
10652	0.099	0740	2	03	01							__
10653	0.082	0760	2	03	01							__
10654	0.912	070F	5	02	42	00	00	00	(CMD_START, SAVE FLASH)			_B___
10655	41.396	0740	2	02	01							__
10656	41.396	0760	2	02	01							__
10657	947.110	070F	1	04		(CMD_END, EXIT BOOTLOADER)						_
10658	0.075	0740	2	04	01							__
10659	0.075	0760	2	04	01							__