**BOOT LOADER**

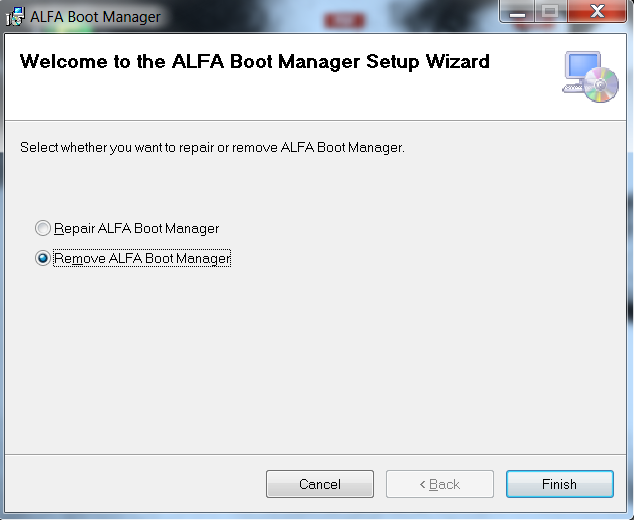
Autore: M.Abelli

data: 20.3.2018

**ALFA USB PROGRAMMER**

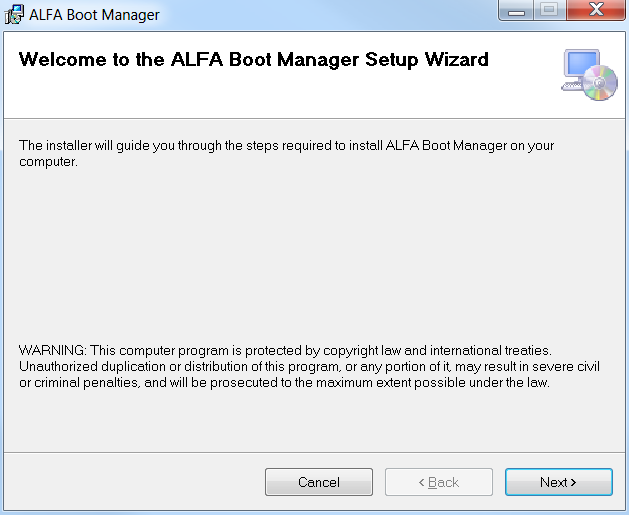
**INSTALLATION PROCEDURE**

1. Run installation File “”
2. If present another Application Program it is necessary before to delete it. So, click on “*Remove ALFA Boot Manager*” option and press “*Finish*” button:



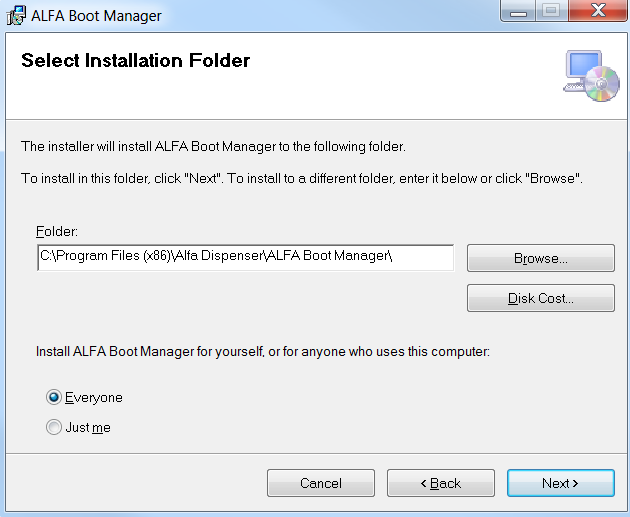
(*Fig. N°1*)

1. Wait to see Installation Complete window and push the button “*Close*”
2. Run again installation File “”. The following window appears:



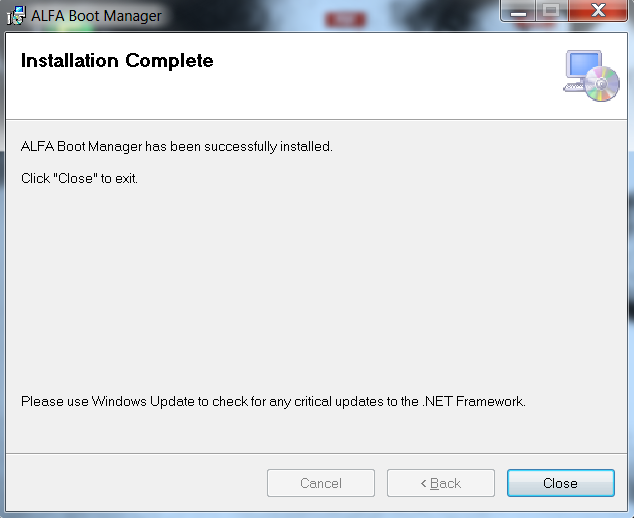
(*Fig. N°2*)

1. Push the button “*Next >*”. In the new window select the folder where the executables File will be installed and check the option “*Everyone*”. Push the button “*Next >*” here and in the next window:



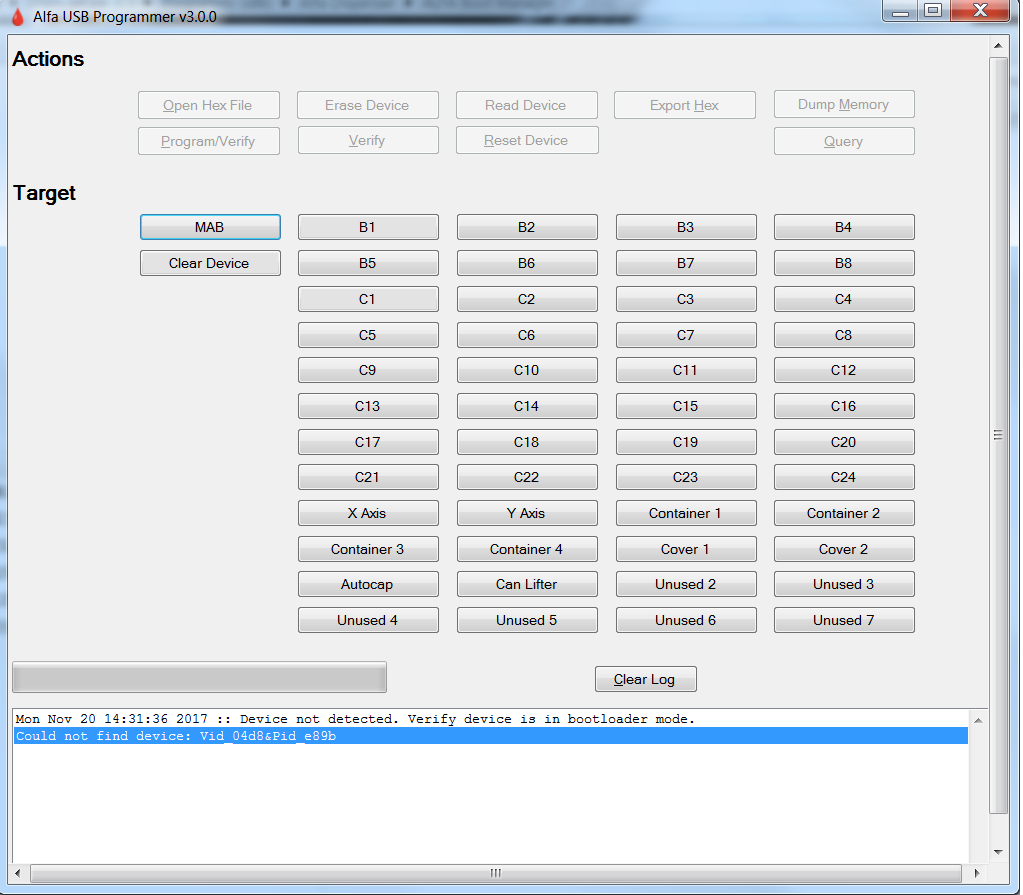
(*Fig. N°3*)

1. Installation is complete now, push the button “*Close*” to exit. Now it is possible to run the application program “*BootLoaderApp*” in the folder selected before



(*Fig. N°4*)

1. To “Run” the program double clicks on “*BootloaderApp*” executable File. No specific libraries are requested:



(*Fig. N°5*)

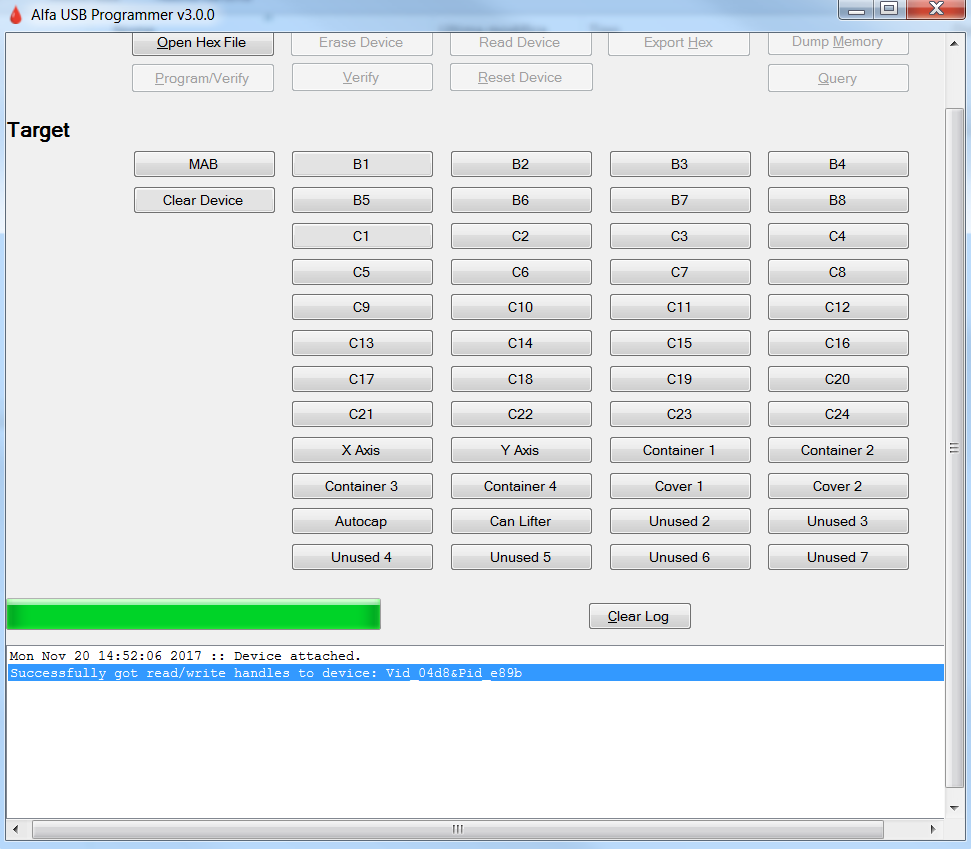
1. “*Alfa USB Programmer v3.0.0.*” is ready to work now

**UPDATING PROCEDURE**

Throughout this program it is possible to update MASTER and ACTUATORS Application program.

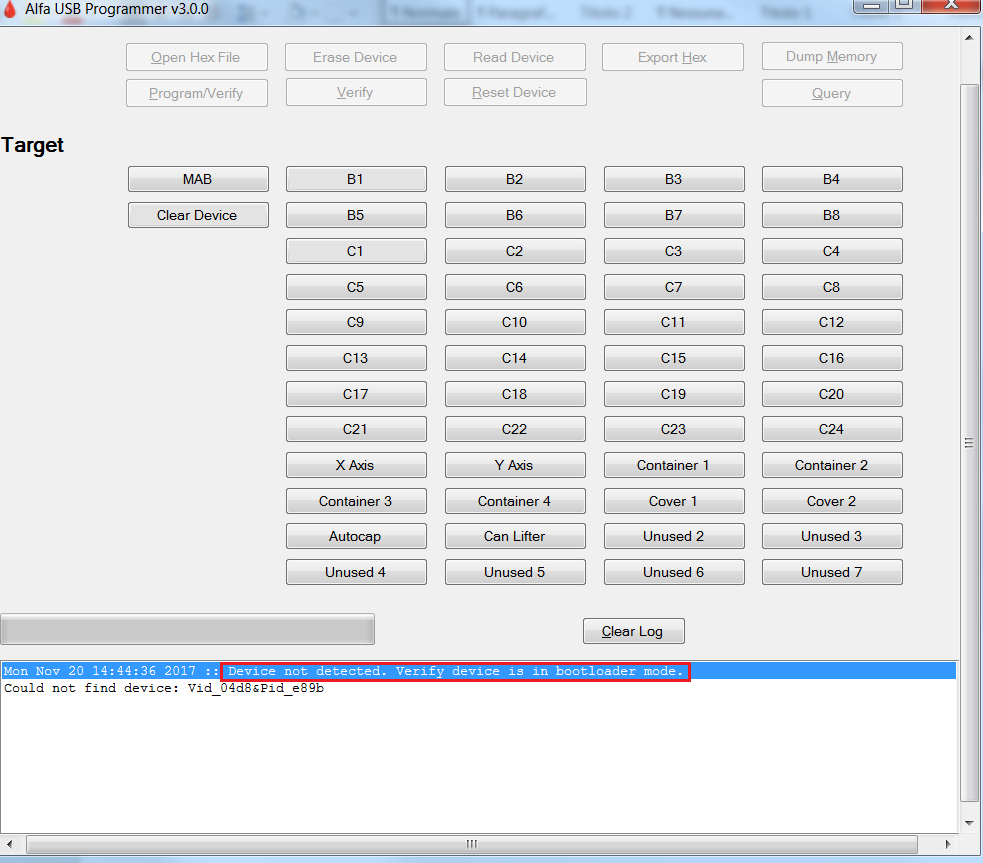
This is the procedure to follow:

1. Connect USB programming cable to MAB board and Laptop
2. Turn ON machine
3. Double clicks on “*BootloaderApp*”.
4. If MASTER BootLoader is correctly connected to USB it appears the message “*Device attached*” and the progress bar is green:



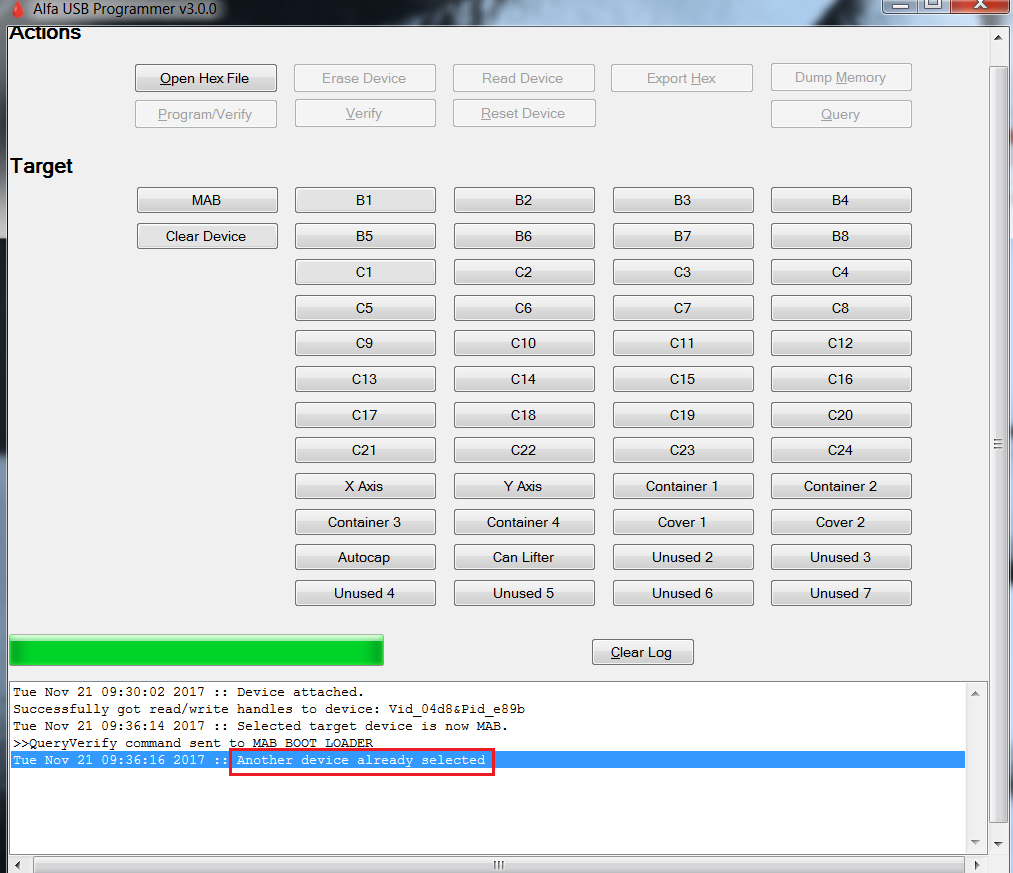
(*Fig. N°6*)

otherwise the message “*Device not detected. Verify device is in bootloader mode*”



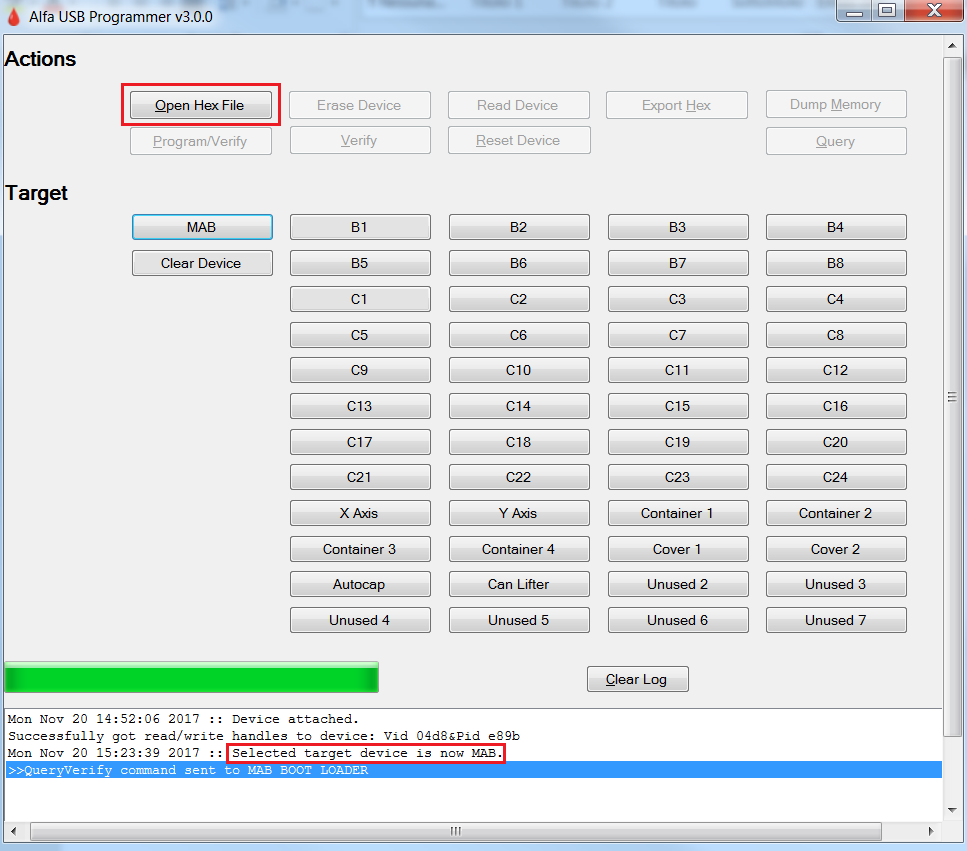
(*Fig. N°7*)

1. In the “*Target*” section, select the ‘*Device ID*’ that we want to update (MAB or ACTUATORS) pressing the corresponding button. In the “Log” section it appears the message “*Selected target device is now*”+’ *Device ID*’ selected. If the ‘*Device ID*’ selected was wrong, before to select the right one, it is necessary to push the clear button “*Clear Device*”. If we try to immediately select a new one we obtain the message “*Another device already selected*”



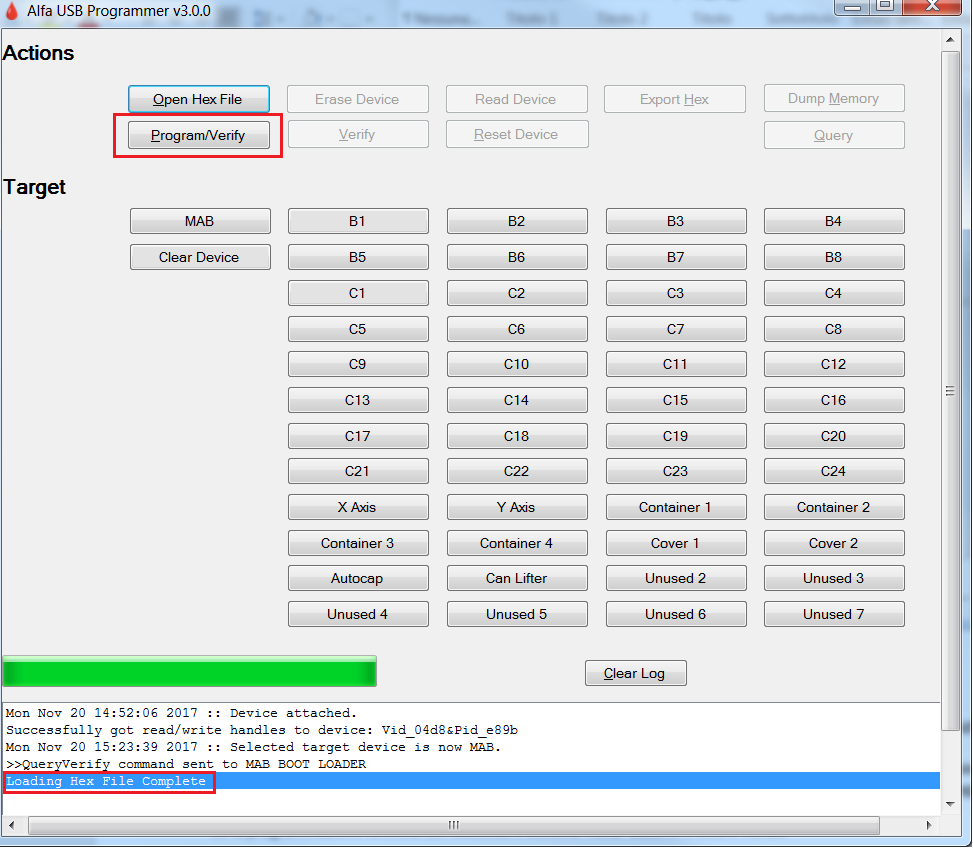
(*Fig. N°8*)

1. Push the button “*Open Hex File*” and select the proper executable file “\*.hex”.



(*Fig. N°9*)

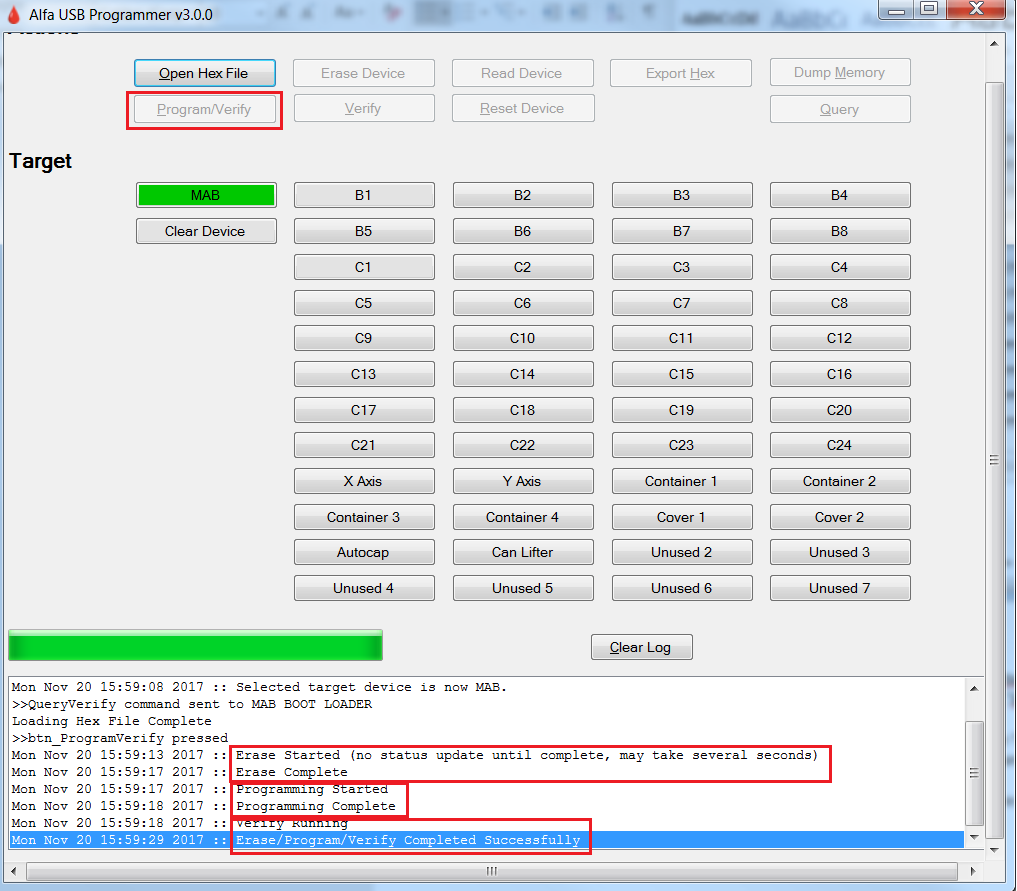
1. Button “*Program/Verify*” becomes now active:



(*Fig. N°10*)

1. Push the button “*Program/Verify*”: the programming sequence is as follow:
2. ERASING - ‘*Device ID*’ application memory is Erased (see the messages in ‘*Log*’ section ‘*Erase Started*’ and ‘*Erase Complete*’)
3. PROGRAMMING - Application program selected is Programmed (see the messages in ‘*Log*’ section ‘*Programming Started*’ and ‘*Programming Complete*’)
4. VERIFY - READ the ‘*Device ID*’ application memory written and compare it with the selected file. Only if there is total matching between the two the procedure ends with success. (see the messages in ‘*Log*’ section ‘*Verify Running*’ and ‘*Erase/Program/Verify Completed Successfully*’)
5. If the procedure ends successfully (progress bar completely green) ‘*Device ID*’ button becomes green.

“*Program/Verify*” button is disabled

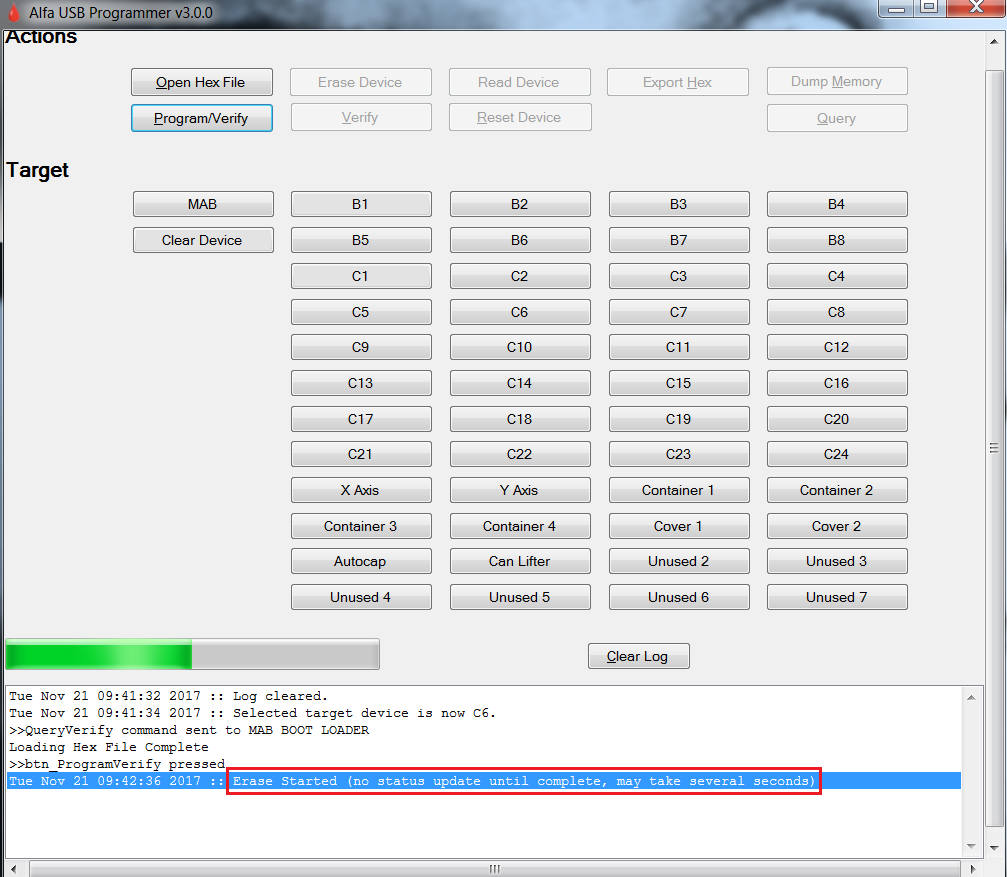


(*Fig. N°11*)

It is NOT possible to update a ‘*Device ID*’ without selecting the proper executable file, throughout “*Open Hex File*” button, immediately after selecting the ‘*Device ID*’.

If we try to update an ACTUATOR without Bootloader, or not connected, or not existing, the procedure will fail at the first step “*Erase Started (no status update until complete, may take several seconds)*”, with the progress bar at 50%. It will be necessary to RESTART:

* Machine
* Alfa USB Programmer.



(*Fig. N°12*)

It is NOT possible to read any Firmware version.

It is NOT possible to make a parallel updating on all Actuators of the same type.

To correctly update machine Firmware it is necessary to Turn ON all the machine altogether.

**PROCESSI**

La procedura di Aggiornamento Firmware di una scheda (MAB o SLAVE) passa attraverso l’implementazione in sequenza dei seguenti processi:

* **QUERY** (richiesta di informazioni di programmazione sulla scheda specificata in cui vuole operare)

Risultati del processo:

*#define QUERY\_IDLE 0xFF*

*#define QUERY\_RUNNING 0x00*

*#define QUERY\_SUCCESS 0x01*

*#define QUERY\_WRITE\_FILE\_FAILED 0x02*

*#define QUERY\_READ\_FILE\_FAILED 0x03*

*#define QUERY\_MALLOC\_FAILED 0x04*

* **ERASE** (cancellazione della memoria FLASH riservata al Firmware Applicativo)

Risultati del processo:

*#define ERASE\_IDLE 0xFF*

*#define ERASE\_RUNNING 0x00*

*#define ERASE\_SUCCESS 0x01*

*#define ERASE\_WRITE\_FILE\_FAILED 0x02*

*#define ERASE\_READ\_FILE\_FAILED 0x03*

*#define ERASE\_VERIFY\_FAILURE 0x04*

*#define ERASE\_POST\_QUERY\_FAILURE 0x05*

*#define ERASE\_POST\_QUERY\_RUNNING 0x06*

*#define ERASE\_POST\_QUERY\_SUCCESS 0x07*

* **PROGRAMMING** (scrittura a blocchi della memoria FLASH riservata al Firmware Applicativo)

Risultati del processo:

*#define PROGRAM\_IDLE 0xFF*

*#define PROGRAM\_RUNNING 0x00*

*#define PROGRAM\_SUCCESS 0x01*

*#define PROGRAM\_WRITE\_FILE\_FAILED 0x02*

*#define PROGRAM\_READ\_FILE\_FAILED 0x03*

*#define PROGRAM\_RUNNING\_ERASE 0x05*

*#define PROGRAM\_RUNNING\_PROGRAM 0x06*

* **VERIFICA** (lettura della memoria FLASH e confronto con il File eseguibile selezionato)

Risultati del processo:

*#define VERIFY\_IDLE 0xFF*

*#define VERIFY\_RUNNING 0x00*

*#define VERIFY\_SUCCESS 0x01*

*#define VERIFY\_WRITE\_FILE\_FAILED 0x02*

*#define VERIFY\_READ\_FILE\_FAILED 0x03*

*#define VERIFY\_MISMATCH\_FAILURE 0x04*

**FILE ESEGUIBILE**

Il file eseguibile da programmare sulla scheda selezionata è generato direttamente dal compilatore MICROCHIP e si presenta in formato Intel HEX.

Per una descrizione del formato di questo file si rimanda a [*https://en.wikipedia.org/wiki/Intel\_HEX*](https://en.wikipedia.org/wiki/Intel_HEX).

Dopo che viene selezionato occorre interpretarlo: da ogni riga si procede ad estrarre l’Indirizzo Assoluto e i Dati da scrivere in memoria FLASH, verificandone la correttezza attraverso il calcolo del checksum che verrà confrontato con quello presente a fine riga.

**COMANDI USB NELLA SEQUENZA DI AGGIORNAMENTO**

Lista di comandi USB da inviare alla macchina:

1. “**QUERY\_DEVICE**” - “**0x02**”: comando usato dall’Host per avere informazioni in merito al device da aggiornare (regioni di memoria di programmare, tipo di memoria). Processo: **QUERY**
2. “**ERASE\_DEVICE**” - “**0x04**”: comando usato per fare partire il processo di ERASE della memoria Applicativo. Il Firmware controlla direttamente quali pagine di memoria si devono cancellare. Processo: **ERASE**
3. “**PROGRAM\_DEVICE**” - “**0x05**”: comando usato per inviare un pacchetto di memoria Flash da programmare. Processo: **PROGRAMMING**
4. “**PROGRAM\_COMPLETE**” – “**0x06**”: comando usato per completare la programmazione della memoria Flash inviando un sottoinsieme residuo di un pacchetto di memoria. Processo: **PROGRAMMING**
5. “**GET\_DATA**” - “**0x07**”: comando inviato al termine della programmazione per leggere una porzione della memoria FLASH. Processo: **VERIFICA**
6. “**JUMP\_TO\_APPLICATION” -** “**0x09**”: comando inviato al termine di tutte le attività di programmazione per fare partire i Programmi Applicativi

**DESCRIZIONE COMANDI USB**

**1.**

**Comando**

**QUERY\_DEVICE**

**Parametri**

*‘****Password[8]*’**: unsigned char - deve coincidere con ‘*mediconPassword[8] = {0x82, 0x14, 0x2A, 0x5D, 0x6F, 0x9A, 0x25, 0x01}’,* altrimenti il comando è rifiutato

**‘*IdFwProgramming*’ = ‘*deviceID*’**: unsigned char - indirizzo della scheda. Se scheda ‘MAB’ = ‘0xFF’

**Outputs**

‘***Command***’: unsigned char - **QUERY\_DEVICE**

‘***PacketDataFieldSize***’: unsigned char - numero di bytes nel campo ‘*Data*’. Valore: ‘0x38’ (=56)

‘***BytesPerAddress***’: unsigned char - numero di bytes per indirizzo . Valore: ‘0x02’

‘***Type1***’: unsigned char - tipo di memoria. Valore: ‘0x01’

‘***Address1***’: unsigned long - indirizzo iniziale di memoria FLASH. Possibili valori:

‘START\_APPL\_ADDRESS\_MASTER’: 0x2C00 (MAB)

‘START\_APPL\_ADDRESS\_SLAVE’: 0x1700 (SLAVE TRANNE UMIDIFICATORE)

‘START\_APPL\_ADDRESS\_HUMIDIFIER’: 0x2000 (UMIDIFICATORE)

‘***Lenght1***’: unsigned long - dimensione della memoria FLASH. Possibili valori.

‘0x27C00’ (MAB)

‘0x4100’ (SLAVE TRANNE UMIDIFICATORE)

‘0x8F00’ (UMIDIFICATORE)

‘***Type2***’: unsigned char - possibili valori:

‘0xFF’: fine della lista dei parametri di memoria

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**2.**

**Comando**

**ERASE\_DEVICE**

**Parametri**

-

**Outputs**

-

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**3.**

**Comando**

**PROGRAM\_DEVICE**

**Parametri**

‘***Address*’**: DWORD - indirizzo di partenza del pacchetto di memoria FLASH da scrivere

‘***Size***’: unsigned char - dimensione in bytes del pacchetto di memoria FLASH da scrivere

**‘*Data[28]*’**: unsigned int - pacchetto di memoria FLASH da scrivere in bytes. Esso contiene 56 bytes.

**Outputs**

-

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**4.**

**Comando**

**PROGRAM\_COMPLETE**

**Parametri**

‘***Address*’**: DWORD - indirizzo di partenza del pacchetto di memoria FLASH da scrivere

‘***Size***’: unsigned char - dimensione in bytes del pacchetto di memoria FLASH da scrivere

**‘*Data[28]*’**: unsigned int - pacchetto di memoria FLASH da scrivere in bytes

**Outputs**

-

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**5.**

**Comando**

**GET\_DATA**

**Parametri**

‘***Address*’**: DWORD - indirizzo di partenza del pacchetto di memoria FLASH da leggere

‘***Size***’: unsigned char - dimensione in bytes del pacchetto di memoria FLASH da leggere

**Outputs**

‘***Address*’**: DWORD - indirizzo di partenza del pacchetto di memoria FLASH letto

‘***Size***’: unsigned char - dimensione in bytes del pacchetto di memoria FLASH letto

**‘*Data[28]*’**: unsigned int - pacchetto di memoria FLASH letto in bytes

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

**6.**

**Comando**

**JUMP\_TO\_APPLICATION**

**Parametri**

-

**Outputs**

-

**SEQUENZA DI OPERAZIONI**

* Selezione della Scheda MAB/Slave da aggiornare
* Selezione del File Intel HEX da aggiornare
* Interpretazione del File Intel HEX
* Invio del comando ‘*QUERY*’, ossia richiesta di informazioni di programmazione sulla scheda specificata in cui vuole operare. Interpretazione della risposta ricevuta
* Invio del comando ‘*ERASE\_DEVICE*’ che fa partire sulla scheda selezionata il processo di ERASE della memoria Applicativo
* Per capire quando il processo di ERASE è terminato occorre inviare periodicamente il comando ‘*QUERY*’: quando si ottiene una risposta dalla macchina significa che l’ERASE è terminato
* Programmazione della memoria FLASH: invio ripetuto del comando ‘*PROGAM\_DEVICE*’ fino a completamento dei blocchi di memoria interi di 56 bytes previsti. Non è prevista una risposta a questo comando
* Terminazione della programmazione della memoria FLASH: invio del comando ’*PROGRAM\_COMPLETE*’ contenente i bytes residui di memoria. Non è prevista una risposta a questo comando
* Processo di VERIFICA di correttezza della programmazione avvenuta. Occorre inviare ripetutamente il comando ‘*GET\_DATA*’ e andare a leggere la risposta che contiene l’indirizzo di partenza del pacchetto di memoria letto, ed il suo contenuto. Si confronterà ciascun pacchetto ricevuto con il corrispondente inviato verificando che coincidano, fino a coprire tutta la memoria riservata al programma Applicativo. Condizione necessaria e sufficiente affinchè la Verica termini positivamente è che tutti i pacchetti previsti letti coincidano con quelli inviati

**DIP SWITCH CONFIGURATION**

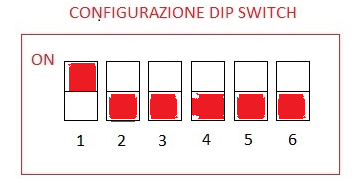
DIP SWITCH configuration follows a binary notation.

Less significant bit is the one at the extremely left side.

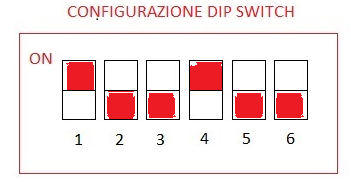
A single dip switch towards ON position corresponds to a bit equal to ‘1’.

SLAVE SCCB1 dip switch setup:

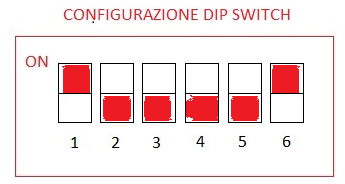
* BASES B1 – B8: ADDRESSES 1..8

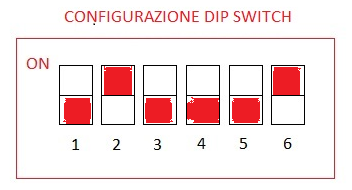
Ex. ADDRESS 1: 

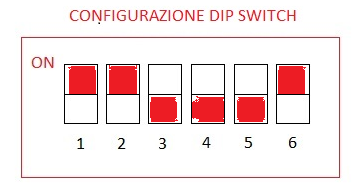
* COLORANTS C1-C24: ADDRESSES 9..32

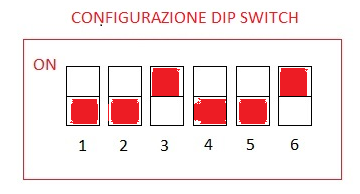
Ex. ADDRESS 9: 

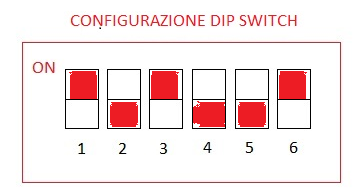
* Remaining SCCB1 follow the addresses codification:

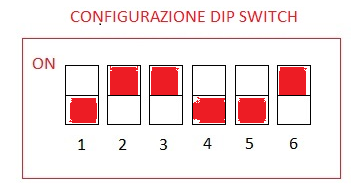
AXIS X: ADDRESS 33 

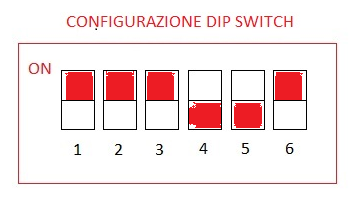
AXIS Y: ADDRESS 34 

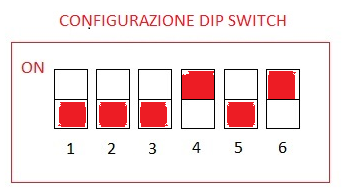
CAN 1: ADDRESS 35 

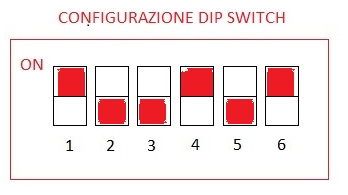
CAN 2: ADDRESS 36 

CAN 3: ADDRESS 37 

CAN 4: ADDERESS 38 

COVER 1 ADDRESS 39 

COVER 2: ADDRESS 40 

AUTOCAP: ADDRESS 41 

CAN LIFTER: ADDRESS 42 