**BOOT LOADER 2.0**

Autore: M.Abelli

data: 14.10.2019

**BOOT LOADER 2.0**

La generazione **2.0** del Boot Loader prevede la possibilità di effettuare l’aggiornamento del Firmware delle macchine ALFA da remoto, unicamente attraverso i Software ALFA, senza dovere spegnere e riaccendere fisicamente la macchina.

L’aggiornamento del Firmware Applicativo avviene sfruttando la connessione USB, Scheda LINUX – MAB (oppure TINTING\_MASTER).

Occorre quindi che le connessioni USB e 232 siano entrambe attive.

Il Software legge la versione Firmware della MAB e delle SLAVE in configurazione macchina (connessione “normale” 232). (comando “**MACHINE\_FW\_VERSIONS**”).

Il Software legge quali SLAVE sono in configurazione macchina (connessione “normale” 232). (comando “**READ\_SLAVES\_CONFIGURATION**”).

Il Software legge la versione Firmware del BootLoader MAB (oppure TINTING\_MASTER) e delle SLAVE in configurazione macchina. (connessione “normale” 232). (comando “**MACHINE\_BOOT\_VERSIONS**”); in tal modo il Software conosce quali SLAVE in configurazione macchina hanno il BootLoader e se si tratta di una versione compatibile con le attività di aggiornamento da remoto. L’assenza del BootLoader sulla scheda MAB (oppure TINTING\_MASTER) preclude qualunque attività di aggiornamento Firmware, l’assenza del BootLoader su una SLAVE preclude l’aggiornamento evidentemente soltanto su di essa.

Qualora venissero riscontrate versioni Firmware più recenti (confronto basato sul nome della versione e non sulla data di compilazione) e se vi fossero le condizioni necessarie per effettuare l’aggiornamento, verrà inviato un messaggio all’operatore che indicherà nome e tipologia delle versioni Firmware trovate, proponendo l’aggiornamento del Firmware.

Possiamo individuare 2 modalità utente per queste attività:

1. **Utente BASE**: l’aggiornamento procede in maniera automatica su tutta la macchina, in maniera SEQUENZIALE scheda per scheda laddove vi siano versioni Firmware più recenti di quelle a bordo macchina
2. **Utente AVANZATO**: può aggiornare anche e non solo il Firmware della singola scheda, anche qualora la versione disponibile fosse più vecchia rispetto a quella a bordo macchina, selezionando manualmente quali versioni intende installare

La condizione di funzionamento “*standard*” è quella di una macchina con attivi tutti i programmi Applicativi, ma che in ciascuna scheda sia presente anche il BootLoader. Se almeno una di queste condizioni viene meno (“*configurazioni non standard*”) si rimanda al paragrafo (‘**GESTIONE CONFIGURAZIONI MACCHINA NON STANDARD’**)

Nel caso sia attivata la procedura di aggiornamento Firmware, si procederà ad inviare alla macchina un comando “**JUMP\_TO\_BOOT**” (connessione “normale” 232) che renderà attivi i Boot Loader presenti.

Un successivo comando di interrogazione della scheda MAB (oppure TINTING\_MASTER) “**BOOT\_FW\_VERSION\_REQUEST**” (connessione USB) permetterà di capire se i Boot Loader sono operativi.

La procedura e la sequenza delle attività di aggiornamento Firmware saranno del tutto analoghe a quelle implementate nel Boot Loader 1.0.

Al termine della procedura di aggiornamento verrà generato un messaggio di “*Update Terminato*”, e sarà possibile attraverso il comando “**JUMP\_TO\_APPLICATION**” (connessione USB) lanciare tutti gli applicativi e quindi tornare al normale funzionamento della macchina.

Il comando di richiesta versione Firmware “**MACHINE\_FW\_VERSIONS**” (connessione “232”) permetterà all’operatore di rendersi conto della correttezza dell’aggiornamento avvenuto.

I Boot Loader delle schede SLAVE rimarranno attivi fintanto che riceveranno dal Boot Loader MAB (oppure TINTING\_MASTER) il comando “**JUMP\_TO\_APPLICATION**” e se verificheranno che un Applicativo è presente in memoria lo lanceranno.

Il Boot Loader della scheda MAB (oppure TINTING\_MASTER) rimarrà sempre attivo tranne quando:

* riceverà dalla scheda Linux, verosimilmente al termine delle operazioni di aggiornamento Firmware, il comando “**JUMP\_TO\_APPLICATION**” (connessione USB) e se verificherà che un Applicativo è presente in memoria
* dopo il power-up allo scadere di un timeout di alcuni secondi, se NON rileverà un precedente comando di “**JUMP\_TO\_BOOT**” (lettura di una locazione di memoria RAM predefinita) e se verificherà che un Applicativo è presente in memoria

**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
7. “**BOOT\_FW\_VERSION\_REQUEST” – “0x0A**”: comando di richiesta della versione di firmware del Bootloader di una SLAVE oppure della MAB (oppure TINTING\_MASTER)

**DESCRIZIONE COMANDI USB LINUX – MAB (TINTING\_MASTER)**

**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’ (oppure TINTING\_MASTER) = ‘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, oppure TINTING\_MASTER)

‘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, oppure TINTING\_MASTER)

‘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**

-

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

**7.**

**Comando**

**BOOT\_FW\_VERSION\_REQUEST**

**Parametri**

***IdFwProgramming*’ = ‘*deviceID*’**: unsigned char - indirizzo della scheda di cui si vuole ottenere la versione di FW BOOT. Se scheda ‘MAB’ (oppure TINTING\_MASTER) )= ‘0xFF’

**Outputs**

‘***Boot\_FW\_Version***’: DWORD – versione di FW del BOOT LOADER, nel formato ‘*0xXXYYZZ’*. Se = ‘*0xFFFFFF*’ il BOOT LOADER è assente

**DESCRIZIONE COMANDI 232 – PROTOCOLLO LINUX – MAB** **(TINTING\_MASTER)**

### 1.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome Variabile e significato** | **Valore** | | **Dimensione (bytes)** |
| **msgType**  Tipo di messaggio | **MACHINE\_FW\_VERSIONS** = 213 | | 1 |
| **MAB\_PROT\_VERSION**  Versione del protocollo di comunicazione MGB – MAB (TINTING\_MASTER) in uso | Numero progressivo. Viene incrementato ogni qualvolta si modifica il protocollo (nuovi comandi, oppure modifica degli esistenti). Valore iniziale: 0x00 | | 1 |
| **MAB\_FW\_VERSION**  versione Firmware MAB (TINTING\_MASTER) | Byte 0  Byte 1  Byte 2 | VERSION\_MAJOR  VERSION\_MINOR  VERSION\_PATCH | 3 |
| **SLAVE\_N1\_FW\_VERSION**  versione Firmware slave N°1 | Byte 0-2 | VERSION\_MAJOR  VERSION\_MINOR  VERSION\_PATCH | 3 |
| ……………………………..….. | ………….…. | ……………….......... | 3 |
| **SLAVE\_N48\_FW\_VERSION**  versione Firmware slave N°48 | Byte 0-2 | VERSION\_MAJOR  VERSION\_MINOR  VERSION\_PATCH | 3 |

### 2.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome Variabile e significato** | **Valore** | | **Dimensione (bytes)** |
| **msgType**  Tipo di messaggio | **MACHINE\_BOOT\_VERSIONS** = 214 | | 1 |
| **MAB\_BOOT\_VERSION**  versione Boot MAB (TINTING\_MASTER) | Byte 0-2 | VERSION\_MAJOR  VERSION\_MINOR  VERSION\_PATCH  Boot NOT present: 0xFFFFFF | 3 |
| **SLAVE\_N1\_BOOT\_VERSION**  versione Boot Slave N°1 | Byte 0-2 | VERSION\_MAJOR  VERSION\_MINOR  VERSION\_PATCH  Boot NOT present: 0xFFFFFF  Slave NOT present: 0x000000 | 3 |
| ……………………………..….. | ………….…. | ……………….......... | 3 |
| **SLAVE\_N48\_BOOT\_VERSION**  versione Boot Slave N°48 | Byte 0-2 | VERSION\_MAJOR  VERSION\_MINOR  VERSION\_PATCH  Boot NOT present: 0xFFFFFF  Slave NOT present: 0x000000 | 3 |

### 3.

|  |  |  |
| --- | --- | --- |
| **Nome Variabile e significato** | **Valore** | **Dimensione (bytes)** |
| **msgType**  Tipo di messaggio | **DIAG\_JUMP\_TO\_BOOT** = 184 | 1 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Nome Variabile e significato** | **Valore** | | **Dimensione (bytes)** |
| **msgType**  Tipo di messaggio | **GET\_SLAVES\_CONFIGURATION** = 215 | | 1 |
| **slaves\_en**  Maschera contenente lo stato di abilitazione di tutte le Slave | Bit 0 | Base n.1: 0 = disabilitato /1 = abilitato | 6 |
| …..... | …........................................................................ |
| Bit 7 | Base n.8: 0 = disabilitato /1 = abilitato |
| Bit 8 | Color n.1: 0 = disabilitato /1 = abilitato |
| …..... | …........................................................................ |
| Bit 31 | Color n.24: 0 = disabilitato /1 = abilitato |
| Bit 32 | Generic actuator n.1: 0 = disabilitato /1 = abilitato |
| …..... | …........................................................................ |
| Bit 47 | Generic actuator n.16: 0 = disabilitato /1 = abilitato |

**DESCRIZIONE COMANDI 485: MAB - ATTUATORI**

|  |  |  |
| --- | --- | --- |
| **Nome Variabile e significato** | **Valore** | **Dimensione (bytes)** |
| **msgType**  Tipo di messaggio | **SALTO\_BOOTLOADER** = 0x15 | 1 |

**DESCRIZIONE COMANDI 485: BOOT MAB – BOOT ATTUATORI**

|  |  |  |
| --- | --- | --- |
| **Nome Variabile e significato** | **Valore** | **Dimensione (bytes)** |
| **msgType**  Tipo di messaggio | **CMD\_FRMWR\_REQUEST** = 0x09 | 1 |

**DESCRIZIONE COMANDI 485: BOOT ATTUATORI - BOOT MAB**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Nome Variabile e significato** | | **Valore** | | | **Dimensione (bytes)** |
| **msgType**  Tipo di messaggio | | **SEND\_FRMWR\_VERSION** = 0x14 | | | 1 |
| **SLAVE\_FW\_VERSION**  versione Firmware slave | | Byte 0-2 | VERSION\_MAJOR  VERSION\_MINOR  VERSION\_PATCH | 3 |

**PROCESSI**

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

* **JUMP TO BOOT**

Processo attivato dall’invio alla MAB (TINTING\_MASTER) dell’omonimo comando “***DIAG\_JUMP\_TO\_BOOT*”**. Il Firmware MAB procede con i seguenti steps:

* Impostazione dello stato macchina “*JUMP\_TO\_BOOT\_ST*” = 9
* Spegnimento del BHL
* Invio del comando di STOP a tutte le SLAVE in configurazione macchina
* Attivazione di un Timer che controlla la Durata del processo di “*STOP*”
* Controllo e verifica che tutte le SLAVE in configurazione macchina siano effettivamente nello stato di “*STOP*”. Se ciò non accade entro un tempo pari a 5” viene generato l’errore “*STOP\_PROCESS\_TIMEOUT\_ERROR*” = 902. La macchina entra nello stato di “*ALARM”,* ed il processo termina qui
* Lettura della versione di BootLoader di tutte le SLAVE in configurazione macchina. Se la MAB (TINTING\_MASTER) NON è provvista di BootLoader che supporta le attività di aggiornamento 2.0 generazione dell’errore “*BOOTLOADER\_MAB\_ABSENT*” = 900
* Invio a tutte le SLAVE in configurazione macchina, aventi un BootLoader con versione appropriata, del comando “*SALTO\_BOOTLOADER*”. Esso sposta l’esecuzione del programma al BootLoader delle SLAVE
* Attivazione di un Timer che controlla la Durata del salto al BootLoader delle SLAVE
* Controllo e verifica che tutte le SLAVE in configurazione macchina abbiano correttamente ricevuto il comando di salto. Se ciò non accade entro un tempo pari a 200msec viene generato l’errore “*FAILED\_JUMP\_TO\_BOOT\_ACTUATOR*” = 903. La macchina entra nello stato di “*ALARM”* per 5” scaduti a quali viene effettuato comunque il salto al BootLoader MAB (TINTING\_MASTER), ed il processo termina qui. Il software ha letto l’errore e consapevole della criticità, quando il BootLoader MAB (TINTING\_MASTER) è attivo, procede ad inviare il comando (connessione USB) “*JUMP\_TO\_APPLICATION*” che fa ripartire tutti i programmi applicativi, tornando quindi nelle condizioni di partenza
* Attesa di 1” affinchè tutti i BootLoader delle SLAVE in configurazione macchina provviste di BootLoader corretti siano operativi
* Impostazione in RAM ad un indirizzo predefinito e condiviso con il BootLoader del valore simbolico ‘*JUMP\_TO\_BOOT\_DONE*’ = 0xAA. Questa informazione verrà utilizzata dal BootLoader per evitare che faccia partire l’applicativo
* Salto al BootLoader della MAB (TINTING\_MASTER) e terminazione del processo. Se malauguratamente il BootLoader MAB (TINTING\_MASTER) non dovesse essere lanciato generazione dell’errore “*FAILED\_JUMP\_TO\_BOOT\_MAB*” = 901. La macchina entra nello stato di “*ALARM*”

Il Software dopo avere inviato il comando “**JUMP TO BOOT”** (connessione “232”) dovrebbe preoccuparsi di vedere che lo stato macchina divenga effettivamente “*JUMP\_TO\_BOOT\_ST*”, verificando anche eventuali errori. A questo punto, dopo una attesa massima di 5”+200msec+1” senza errori, dovrebbe inviare il comando “***BOOT\_FW\_VERSION\_REQUEST*”** (connessione “USB”) rivolto alla MAB (oppure TINTING\_MASTER). Se riceve una risposta corretta (vedere descrizione del comando), allora considera il processo completato senza errori.

* **BOOT FW VERSION REQUEST**

Comando inviato al Boot Loader MAB (oppure TINTING\_MASTER) (connessione USB) dell’omonimo comando “***BOOT\_FW\_VERSION\_REQUEST***”. Si procede nel seguente modo:

* Attraverso il parametro‘***deviceID***’ si vuole leggere la versione di Boot Loader della MAB (oppure TINTING\_MASTER) (= 0xFF). Una risposta positiva al comando valida positivamente il processo di “**JUMP TO BOOT**”.
* Successivamente si invia il comando specificando l’indirizzo di tutte e sole le Slave in configurazione macchina. Questa informazione è stata ottenuta in precedenza inviando all’Applicativo MAB (oppure TINTING\_MASTER), il comando “***READ\_SLAVES\_CONFIGURATION***”. Se una Slave non risponde al comando (sconnessa, assenza di programma) il Boot Loader MAB (oppure TINTING\_MASTER) invia la risposta 0xFFFFFF dopo 100msec.
* **QUERY** (richiesta di informazioni di programmazione sulla scheda specificata in cui vuole operare) (connessione USB)
* **ERASE** (cancellazione della memoria FLASH riservata al Firmware Applicativo) (connessione USB)
* **PROGRAMMING** (scrittura a blocchi della memoria FLASH riservata al Firmware Applicativo) (connessione USB)
* **VERIFICA** (lettura della memoria FLASH e confronto con il File eseguibile selezionato) (connessione USB)
* **JUMP TO APPLICATION**

Processo attivato dall’invio al Boot Loader MAB (oppure TINTING\_MASTER) (connessione USB) dell’omonimo comando “***JUMP\_TO\_APPLICATION*”**. Il Boot Loader MAB (oppure TINTING\_MASTER) procede con i seguenti steps:

* Controllo che nella MAB (oppure TINTING\_MASTER) sia presente un Applicativo attraverso la lettura di una locazione di memoria RAM predefinita
* Se è presente un Applicativo invio di un comando speciale “Broadcast” a tutte le schede Slave connesse: quelle che hanno attivo il Boot Loader se e solo se hanno anche un programma Applicativo procederanno a lanciarlo
* Trascorso un tempo pari a 2 sec il Boot Loader MAB (oppure TINTING\_MASTER) lancia il suo programma Applicativo
* Dopo 5 sec dall’invio del comando “***JUMP\_TO\_APPLICATION*”** si invia Il comando di richiesta versione Firmware “**MACHINE\_FW\_VERSIONS**” (connessione “232”) che permetterà all’operatore di rendersi conto della correttezza dell’aggiornamento avvenuto.

**SEQUENZA DI OPERAZIONI**

* Selezione della Scheda MAB(TINTING\_MASTER) /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

**VERSIONI FIRMWARE**

**VERSIONE FIRMWARE**: X.Y.Z

|  |  |  |
| --- | --- | --- |
|  | **BOOT** | **APPLICATIVO BOOT COMPATIBILE** |
| **MAB (DESK, COLOR LAB)** | **1**.**0**.2 | ≥ **3**.2.0 |
| **MAB (THOR)** | **1**.**0**.2 | ≥ **4**.1.0 |
| **TINTING MASTER** | **1.9**.2 | ≥ **4**.1.0 |
| **SLAVE (DESK, COLOR LAB)** | **1**.**5**.2 | ≥ **4**.2.0 |
| **ATTUATORE GRUPPO DOPPIO (THOR)** | **1**.**5**.2 | ≥ **4**.2.0 |
| **COLORANTE SINGOLO (THOR)** | **1**.**5**.2 | ≥ **4**.2.0 |
| **AUTOCAP (THOR)** | **1**.**5**.2 | ≥ **4**.2.0 |
| **CAN LIFTER** | **1**.**6**.2 | ≥ **3**.2.0 |
| **HUMIDIFIER** | **1**.**7**.2 | ≥ **3**.2.0 |
| **TINTING SLAVE** | **1.8**.2 | ≥ **4**.1.0 |

**NUOVE ATTIVITA’ FIRMWARE DA IMPLEMENTARE**

**BOOT MAB DESK + THOR 1.0**

1. Comando “**JUMP\_TO\_APPLICATION**”
2. Comando di richiesta Versione Firmware BootLoader di se stesso e di una SLAVE “**BOOT\_FW\_VERSION\_REQUEST**”
3. Lettura da indirizzo RAM predefinito di eventuale comando “**DIAG\_JUMP\_TO\_BOOT**” inviato in precedenza

“**BOOT\_FW\_VERSION\_REQUEST**” presente, ma manca la comunicazione con le Slave

“**JUMP\_TO\_APPLICATION**” implementato

**BOOT TINTING MASTER**

1. Comando “**JUMP\_TO\_APPLICATION**”
2. Comando di richiesta Versione Firmware BootLoader di se stesso e di una SLAVE “**BOOT\_FW\_VERSION\_REQUEST**”
3. Lettura da indirizzo RAM predefinito di eventuale comando “**DIAG\_JUMP\_TO\_BOOT**” inviato in precedenza

**APPLICATIVO MAB DESK**

1. Comando “**MACHINE\_BOOT\_VERSIONS”**
2. Comando **“DIAG\_JUMP\_TO\_BOOT”** e relativo processo

“**MACHINE\_BOOT\_VERSIONS**” presente, ma manca la comunicazione con le Slave

**“DIAG\_JUMP\_TO\_BOOT”** processo implementato

**APPLICATIVO MAB THOR 1.0**

1. Comando “**MACHINE\_BOOT\_VERSIONS”**
2. Comando **“DIAG\_JUMP\_TO\_BOOT”** e relativo processo

**APPLICATIVO TINTING MASTER**

1. Comando “**MACHINE\_BOOT\_VERSIONS”**
2. Comando **“DIAG\_JUMP\_TO\_BOOT”** e relativo processo

**BOOT SLAVE DESK + THOR 1.0 + THOR 2.0**

1. Salvataggio della Versione Firmware BootLoader nella locazione di memoria RAM condivisa con i rispettivi Applicativi
2. Comando di richiesta Versione Firmware dal Boot Loader MAB

Verificato presente nel Bootloader della SCCB e della TINTING

**APPLICATIVO SLAVE DESK**

1. Gestione della Versione Firmware BootLoader (nuovo campo nel comando “**CONTROLLO\_PRESENZA**”, lettura da indirizzo in memoria RAM condiviso con il rispettivo BootLoader)
2. Comando **“SALTO\_BOOTLOADER”**

Applicativo Tinting: manca il nuovo campo nel comando **“CONTROLLO\_PRESENZA”.** E’ presente invece il comando e il salto al bootloader.

Applicativo Slave: manca il nuovo campo nel comando **“CONTROLLO\_PRESENZA”.** E’ presente invece il comando e il salto al bootloader.

**APPLICATIVO SLAVE THOR 1.0 + THOR 2.0**

1. Gestione della Versione Firmware BootLoader (nuovo campo nel comando “**CONTROLLO\_PRESENZA**”, lettura da indirizzo in memoria RAM condiviso con il rispettivo BootLoader)
2. Comando **“SALTO\_BOOTLOADER”**

**THOR 2.0**

**BOOT TINTING MASTER OK**

1. Comando “**JUMP\_TO\_APPLICATION**”

Jump\_To\_Application\_Request();

1. Comando di richiesta Versione Firmware BootLoader di se stesso e di una SLAVE “**BOOT\_FW\_VERSION\_REQUEST**”

Master\_Boot\_FW\_Request()

\_\_APPL\_DATA\_BASE (0x1010)

\_\_BL\_VERSION\_ADDR (\_\_APPL\_DATA\_BASE - 0x18)

boot\_fw\_version \_\_attribute\_\_((space(data), address(\_\_BL\_VERSION\_ADDR)));

Boot\_FW\_Request()

BL\_SLAVE\_VERSION

1. Lettura da indirizzo RAM predefinito di eventuale comando “**DIAG\_JUMP\_TO\_BOOT**” inviato in precedenza

\*PtrJMPBoot = (unsigned long \*) (\_\_JMP\_BOOT\_ADDR);

\_\_APPL\_DATA\_BASE (0x1010)

\_\_JMP\_BOOT\_ADDR (\_\_APPL\_DATA\_BASE - 0x1C)

**APPLICATIVO TINTING MASTER OK**

1. Comando “**MACHINE\_BOOT\_VERSIONS”**

\_\_APPL\_DATA\_BASE 0x1010

\_\_BL\_SW\_VERSION (\_\_APPL\_DATA\_BASE - 0x18)

#define BOOT\_FW\_VERSION() (\*(BootPtrTestResults)): lettura di una locazione RAM predefinita sulla MMT

slaves\_boot\_versions: ottenuto dal comando di Stato delle Slave

1. Comando **“DIAG\_JUMP\_TO\_BOOT”** e relativo processo

case COLOR\_JUMP\_TO\_BOOT:

colorAct[slave\_id].colorFlags.jump\_to\_boot = TRUE;

set\_slave\_status(B1\_BASE\_IDX + slave\_id, 0);

break;

\_\_APPL\_DATA\_BASE 0x1010

// This location is used to store Boot Loader Firmware Version

\_\_JMP\_BOOT\_ADDR (\_\_APPL\_DATA\_BASE - 0x1C)

**BOOT SLAVE OK**

1. Salvataggio della Versione Firmware BootLoader nella locazione di memoria RAM condivisa con i rispettivi Applicativi

\_\_APPL\_DATA\_BASE (0x9E0)

\_\_BL\_SW\_VERSION (\_\_APPL\_DATA\_BASE - 0x6)

boot\_fw\_version \_\_attribute\_\_((space(data), address(\_\_BL\_SW\_VERSION)));

boot\_fw\_version = (unsigned long) BL\_SW\_VERSION;

1. Comando di richiesta Versione Firmware dal Boot Loader MAB

Comando implementato

**APPLICATIVO SLAVE OK**

1. Gestione della Versione Firmware BootLoader (nuovo campo nel comando “**CONTROLLO\_PRESENZA**”, lettura da indirizzo in memoria RAM condiviso con il rispettivo BootLoader)

Versione presente nel comando “**CONTROLLO\_PRESENZA**”

BOOT\_FW\_VERSION() (\*(BootPtrTestResults))

const unsigned long \*BootPtrTestResults = (unsigned long \*) (\_\_BL\_SW\_VERSION);

\_\_APPL\_DATA\_BASE 0x9F0

\_\_BL\_SW\_VERSION (\_\_APPL\_DATA\_BASE - 0x16)

1. Comando **“SALTO\_BOOTLOADER”**

if ((Status.level == COLOR\_JUMP\_TO\_BOOT\_ST) && (Start\_Jump\_Boot == 1)) {

jump\_to\_boot();

jump\_to\_boot()

\_\_asm\_\_ volatile ("goto " \_\_BOOT\_GOTO\_ADDR);

\_\_BOOT\_GOTO\_ADDR "0x0204"

Nessun controllo che il Boot sia effettivamente presente e con la versione giusta

Ore stimate per le attività di sviluppo: 120

**GESTIONE CONFIGURAZIONI MACCHINA NON STANDARD**

**CAVI SCOLLEGATI**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | USB  COLLEGATO | USB  SCOLLEGATO | 232  COLLEGATO | 232  SCOLLEGATO | 485  COLLEGATO | 485  SCOLLEGATO |
| MAB | X |  |  | X |  |  |
| SLAVE 1 |  |  |  |  | X |  |
| SLAVE N |  |  |  |  | X |  |

Dispatcher manda comando di Richiesta versione FW APPLICATIVO, non ottiene risposta xchè il cavo 232 è scollegato.

Prova a mandare via USB il comando di richiesta versione FW BOOT al BOOT MAB. Se ottiene risposta significa che è attivo il BOOT.

Si provvede a inviare lo stesso comando FW BOOT a tutte le SLAVE possibili.

Dispatcher procede ad aggiornare la MAB e le sole SLAVE da cui ha ottenuto la versione FW BOOT.

Al termine manda il comando JUMP\_TO\_APP. Essendo scollegato il cavo 232 non riuscirà ad usare la macchina.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | USB  COLLEGATO | USB  SCOLLEGATO | 232  COLLEGATO | 232  SCOLLEGATO | 485  COLLEGATO | 485  SCOLLEGATO |
| MAB | X |  |  | X |  |  |
| SLAVE 1 |  |  |  |  | X |  |
| SLAVE N |  |  |  |  |  | X |

Dispatcher manda comando di Richiesta versione FW APPLICATIVO, non ottiene risposta xchè il cavo 232 è scollegato.

Prova a mandare via USB il comando di richiesta versione FW BOOT al BOOT MAB. Se ottiene risposta significa che è attivo il BOOT.

Si provvede a inviare lo stesso comando FW BOOT a tutte le SLAVE possibili.

Dispatcher procede ad aggiornare la MAB ed eventualmente alla sola SLAVE 1 se ha ottenuto la versione FW BOOT.

Al termine manda il comando JUMP\_TO\_APP. Essendo scollegato il cavo 232 non riuscirà ad usare la macchina.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | USB  COLLEGATO | USB  SCOLLEGATO | 232  COLLEGATO | 232  SCOLLEGATO | 485  COLLEGATO | 485  SCOLLEGATO |
| MAB | X |  |  | X |  |  |
| SLAVE 1 |  |  |  |  |  | X |
| SLAVE N |  |  |  |  |  | X |

Dispatcher manda comando di Richiesta versione FW APPLICATIVO, non ottiene risposta xchè il cavo 232 è scollegato.

Prova a mandare via USB il comando di richiesta versione FW BOOT al BOOT MAB. Se ottiene risposta significa che è attivo il BOOT.

Si provvede a inviare lo stesso comando FW BOOT a tutte le SLAVE possibili.

Dispatcher procede ad aggiornare solo la MAB, ma nessuna SLAVE dal momento che il cavo 485 è scollegato.

Al termine manda il comando JUMP\_TO\_APP. Essendo scollegato il cavo 232 non riuscirà ad usare la macchina.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | USB  COLLEGATO | USB  SCOLLEGATO | 232  COLLEGATO | 232  SCOLLEGATO | 485  COLLEGATO | 485  SCOLLEGATO |
| MAB | X |  | X |  |  |  |
| SLAVE 1 |  |  |  |  | X |  |
| SLAVE N |  |  |  |  |  | X |

L’aggiornamento procede normalmente, tranne per il fatto che il comando di richiesta FW BOOT sulla SLAVE N non ha risposta dal momento che il cavo 485 è scollegato. Anche un precedente eventuale comando JUMP\_TO\_BOOT su questa scheda non ha avuto effetto. Al termine la macchina sarà aggiornata ma data la sconnessione della 485 ci sarà un errore di Timeout con la SLAVE N.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | USB  COLLEGATO | USB  SCOLLEGATO | 232  COLLEGATO | 232  SCOLLEGATO | 485  COLLEGATO | 485  SCOLLEGATO |
| MAB | X |  | X |  |  |  |
| SLAVE 1 |  |  |  |  |  | X |
| SLAVE N |  |  |  |  |  | X |

L’aggiornamento procede normalmente, tranne per il fatto che il comando di richiesta FW BOOT sulle SLAVE non ha risposta dal momento che il cavo 485 è scollegato. Anche un precedente eventuale comando JUMP\_TO\_BOOT sulle SLAVE non ha avuto effetto. Al termine la macchina sarà aggiornata ma data la sconnessione della 485 ci sarà un errore di Timeout su tutte le SLAVE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | USB  COLLEGATO | USB  SCOLLEGATO | 232  COLLEGATO | 232  SCOLLEGATO | 485  COLLEGATO | 485  SCOLLEGATO |
| MAB |  | X |  | X |  |  |
| SLAVE 1 |  |  |  |  | X |  |
| SLAVE N |  |  |  |  | X |  |

Nessuna operazione è possibile essendo entrambi i cavi USB e 232 scollegati

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | USB  COLLEGATO | USB  SCOLLEGATO | 232  COLLEGATO | 232  SCOLLEGATO | 485  COLLEGATO | 485  SCOLLEGATO |
| MAB |  | X |  | X |  |  |
| SLAVE 1 |  |  |  |  | X |  |
| SLAVE N |  |  |  |  |  | X |

Nessuna operazione è possibile essendo entrambi i cavi USB e 232 scollegati

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | USB  COLLEGATO | USB  SCOLLEGATO | 232  COLLEGATO | 232  SCOLLEGATO | 485  COLLEGATO | 485  SCOLLEGATO |
| MAB |  | X |  | X |  |  |
| SLAVE 1 |  |  |  |  |  | X |
| SLAVE N |  |  |  |  |  | X |

Nessuna operazione è possibile essendo entrambi i cavi USB e 232 scollegati

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | USB  COLLEGATO | USB  SCOLLEGATO | 232  COLLEGATO | 232  SCOLLEGATO | 485  COLLEGATO | 485  SCOLLEGATO |
| MAB |  | X | X |  |  |  |
| SLAVE 1 |  |  |  |  | X |  |
| SLAVE N |  |  |  |  |  | X |

Se è presente il BOOT MAB è possibile mandare il comando JUMP\_TO\_BOOT che farà partire anche il BOOT della SLAVE 1 se presente. Il Problema è che essendo scollegato il cavo USB non si potrà effettuare nessuna operazione e la macchina resterà sempre così con il BOOT attivo.

**Bisognerebbe che il Firmware MAB dell’Applicativo si accorgesse che il cavo USB è connesso. Ma non avremmo spazio in memoria FLASH sulla TINTING MASTER**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | USB  COLLEGATO | USB  SCOLLEGATO | 232  COLLEGATO | 232  SCOLLEGATO | 485  COLLEGATO | 485  SCOLLEGATO |
| MAB |  | X | X |  |  |  |
| SLAVE 1 |  |  |  |  |  | X |
| SLAVE N |  |  |  |  |  | X |

Se è presente il BOOT MAB è possibile mandare il comando JUMP\_TO\_BOOT che farà partire solo il BOOT MAB essendo il cavo 485 scollegato per entrambe le SLAVE. Il Problema è che essendo scollegato il cavo USB non si potrà effettuare nessuna operazione e la macchina resterà sempre così con il BOOT attivo.

**--------------------------------------------------------------------------------------------------------------------------------**

**N°1**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB | X |  |  |
| SLAVE 1 | X |  |  |
| SLAVE N | X |  |  |

Boot MAB NON fa il salto xchè manca Applicativo: a causa di questo anche tutte le Slave rimangono con attivo il BOOT.

Dispatcher manda comando di Richiesta versione FW APPLICATIVO, non ottiene risposta per cui prova a mandare via USB il comando di richiesta versione FW BOOT al BOOT MAB. Ottiene risposta per cui provvede a inviare lo stesso comando FW BOOT a tutte le SLAVE possibili.

Dispatcher procede ad aggiornare SLAVE1, SLAVEN, al termine MAB (l’ordine non è importante) e al termine manda il comando JUMP\_TO\_APP.

**N°2**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB | X |  |  |
| SLAVE 1 | X |  |  |
| SLAVE N |  | X |  |

Boot MAB NON fa il salto xchè manca Applicativo: a causa di questo anche tutte le Slave rimangono con attivo il BOOT.

Dispatcher manda comando di Richiesta versione FW APPLICATIVO, non ottiene risposta per cui prova a mandare via USB il comando di richiesta versione FW BOOT al BOOT MAB. Ottiene risposta per cui provvede a inviare lo stesso comando FW BOOT a tutte le SLAVE possibili. Dispatcher ottiene la risposta 0xFF dalla SLAVE N, il che indica che NON vi è un BOOT attivo.

Dispatcher procede ad aggiornare SOLO SLAVE1, al termine MAB (l’ordine non è importante) e al termine manda il comando JUMP\_TO\_APP.

**N°3**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB | X |  |  |
| SLAVE 1 | X |  |  |
| SLAVE N |  |  | X |

SLAVEN: se attivo il BOOT si rientra nel caso N°1, se attivo APPLICATIVO si rientra nel caso N°2

**N°4**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB | X |  |  |
| SLAVE 1 |  | X |  |
| SLAVE N |  | X |  |

Boot MAB NON fa il salto xchè manca Applicativo: a causa di questo anche tutte le Slave rimangono con attivo il BOOT.

Dispatcher manda comando di Richiesta versione FW APPLICATIVO, non ottiene risposta per cui prova a mandare via USB il comando di richiesta versione FW BOOT al BOOT MAB. Ottiene risposta SOLO dalla MAB.

Dispatcher procede ad aggiornare SOLO il F

irmware MAB e al termine manda il comando JUMP\_TO\_APP.

**N°5**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB | X |  |  |
| SLAVE 1 |  | X |  |
| SLAVE N |  |  | X |

Boot MAB NON fa il salto xchè manca Applicativo: a causa di questo anche tutte le Slave rimangono con attivo il BOOT.

Dispatcher manda comando di Richiesta versione FW APPLICATIVO, non ottiene risposta per cui prova a mandare via USB il comando di richiesta versione FW BOOT al BOOT MAB. Ottiene risposta dalla MAB e dalla SLAVE N se attivo il BOOT.

Dispatcher procede ad aggiornare il firmware SLAVE N, se è attivo il BOOT, e quello della MAB e al termine manda il comando JUMP\_TO\_APP.

**N°6**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB | X |  |  |
| SLAVE 1 |  |  | X |
| SLAVE N |  |  | X |

SLAVE1: se attivo il BOOT si rientra nel caso N°1, se attivo APPLICATIVO si rientra nel caso N°2

SLAVEN: se attivo il BOOT si rientra nel caso N°1, se attivo APPLICATIVO si rientra nel caso N°2

**N°7**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  | X |  |
| SLAVE 1 | X |  |  |
| SLAVE N | X |  |  |

Fw MAB NON esegue il comando JUMP\_TO\_BOOT xchè manca il BOOT MAB.

Nessuna attività di aggiornamento possibile.

Macchina in errore xchè SLAVE 1 e SLAVE N non hanno l’Applicativo.

**N°8**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  | X |  |
| SLAVE 1 | X |  |  |
| SLAVE N |  | X |  |

Fw MAB NON esegue il comando JUMP\_TO\_BOOT xchè manca il BOOT MAB.

Nessuna attività di aggiornamento possibile.

Macchina in errore xchè SLAVE 1 non ha l’Applicativo.

**N°9**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  | X |  |
| SLAVE 1 | X |  |  |
| SLAVE N |  |  | X |

Fw MAB NON esegue il comando JUMP\_TO\_BOOT xchè manca il BOOT MAB.

Nessuna attività di aggiornamento possibile.

Macchina in errore xchè SLAVE 1 non ha l’Applicativo.

**N°10**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  | X |  |
| SLAVE 1 |  | X |  |
| SLAVE N |  | X |  |

Fw MAB NON esegue il comando JUMP\_TO\_BOOT xchè manca il BOOT MAB.

Nessuna attività di aggiornamento possibile.

Macchina funzionante correttamente.

**N°11**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  | X |  |
| SLAVE 1 |  | X |  |
| SLAVE N |  |  | X |

Fw MAB NON esegue il comando JUMP\_TO\_BOOT xchè manca il BOOT MAB.

Nessuna attività di aggiornamento possibile.

Macchina funzionante correttamente.

**N°12**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  | X |  |
| SLAVE 1 |  |  | X |
| SLAVE N |  |  | X |

Fw MAB NON esegue il comando JUMP\_TO\_BOOT xchè manca il BOOT MAB.

Nessuna attività di aggiornamento possibile.

Macchina funzionante correttamente.

**--------------------------------------------------------------------------------------------------------------------------------**

**N°13**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  |  | X |
| SLAVE 1 | X |  |  |
| SLAVE N | X |  |  |

1. Fw MAB con attivo l’Applicativo.

Fw MAB esegue il comando JUMP\_TO\_BOOT xchè è presente il BOOT MAB.

Dispatcher manda via USB il comando di richiesta versione FW BOOT al BOOT MAB. Ottiene risposta per cui provvede a inviare lo stesso comando FW BOOT a tutte le SLAVE possibili e da esse ha risposta positiva.

Dispatcher procede ad aggiornare SLAVE1, SLAVEN, al termine MAB (l’ordine non è importante) e al termine manda il comando JUMP\_TO\_APP.

1. Fw MAB con attivo il Boot: vedi caso N°1

**N°14**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  |  | X |
| SLAVE 1 | X |  |  |
| SLAVE N |  | X |  |

1. Fw MAB con attivo l’Applicativo.

Fw MAB esegue il comando JUMP\_TO\_BOOT xchè è presente il BOOT MAB.

Dispatcher manda via USB il comando di richiesta versione FW BOOT al BOOT MAB. Ottiene risposta per cui provvede a inviare lo stesso comando FW BOOT a tutte le SLAVE possibili e da esse ha risposta positiva SOLO da SLAVE 1

Dispatcher procede ad aggiornare SLAVE1, al termine MAB (l’ordine non è importante) e al termine manda il comando JUMP\_TO\_APP. La macchina funzionerà correttamente.

1. Fw MAB con attivo il Boot: vedi caso N°2

**N°15**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  |  | X |
| SLAVE 1 | X |  |  |
| SLAVE N |  |  | X |

1. Fw MAB con attivo l’Applicativo.

Fw MAB esegue il comando JUMP\_TO\_BOOT xchè è presente il BOOT MAB.

Dispatcher manda via USB il comando di richiesta versione FW BOOT al BOOT MAB. Ottiene risposta per cui provvede a inviare lo stesso comando FW BOOT a tutte le SLAVE possibili e da esse ha risposta positiva da SLAVE 1 e da SLAVE N.

Dispatcher procede ad aggiornare SLAVE1 e SLAVE N, al termine MAB (l’ordine non è importante) e al termine manda il comando JUMP\_TO\_APP. La macchina funzionerà correttamente.

1. Fw MAB con attivo il Boot: vedi caso N°3

**N°16**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  |  | X |
| SLAVE 1 |  | X |  |
| SLAVE N |  | X |  |

1. Fw MAB con attivo l’Applicativo.

Fw MAB esegue il comando JUMP\_TO\_BOOT xchè è presente il BOOT MAB.

Dispatcher manda via USB il comando di richiesta versione FW BOOT al BOOT MAB. Ottiene risposta per cui provvede a inviare lo stesso comando FW BOOT a tutte le SLAVE possibili e da nessuna di esse ha risposta positiva.

Dispatcher procede ad aggiornare SOLO la MAB e al termine manda il comando JUMP\_TO\_APP. La macchina funzionerà correttamente.

1. Fw MAB con attivo il Boot: vedi caso N°4

**N°17**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  |  | X |
| SLAVE 1 |  | X |  |
| SLAVE N |  |  | X |

1. Fw MAB con attivo l’Applicativo.

Fw MAB esegue il comando JUMP\_TO\_BOOT xchè è presente il BOOT MAB.

Dispatcher manda via USB il comando di richiesta versione FW BOOT al BOOT MAB. Ottiene risposta per cui provvede a inviare lo stesso comando FW BOOT a tutte le SLAVE possibili e ha risposta positiva da SLAVE N.

Dispatcher procede ad aggiornare SLAVE N e al termine MAB (l’ordine non è importante) e al termine manda il comando JUMP\_TO\_APP. La macchina funzionerà correttamente.

1. Fw MAB con attivo il Boot: vedi caso N°5

**N°18**

|  |  |  |  |
| --- | --- | --- | --- |
|  | BOOT | APPLICATIVO | BOOT + APPLICATIVO |
| MAB |  |  | X |
| SLAVE 1 |  |  | X |
| SLAVE N |  |  | X |

1. Fw MAB con attivo l’Applicativo.

Fw MAB esegue il comando JUMP\_TO\_BOOT xchè è presente il BOOT MAB.

Dispatcher manda via USB il comando di richiesta versione FW BOOT al BOOT MAB. Ottiene risposta per cui provvede a inviare lo stesso comando FW BOOT a tutte le SLAVE possibili e ha risposta positiva da tutte le SLAVE.

Dispatcher procede ad aggiornare SLAVE 1, SLAVE N e al termine MAB (l’ordine non è importante) e al termine manda il comando JUMP\_TO\_APP. La macchina funzionerà correttamente.

1. Fw MAB con attivo il Boot: vedi caso N°6