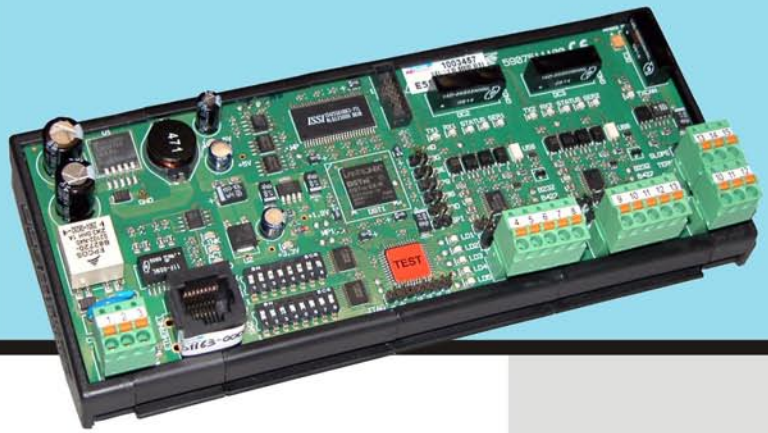
Manuale uso e manutenzione



Bridge Ethernet CAN Seriale

Codice ordine: **1902101002**

Data: **Ottobre 2009**

Rev: **1.2**

Sommario

SYSTEM Electronics

CIA

Member of CBITBUS European Users Group

Storico Revisioni			Pagine
Rev.	1.0	Stesura	9
Rev.	1.1	Modifica Tabella 2.2	9
Rev.	1.2	Aggiunto Figura 2.1.2	9

AZIENDA CON SISTEMA DI GESTIONE
PER LA QUALITA' CERTIFICATO DA DNV
= UNI EN ISO 9001:2000 =

SYSTEM s.p.a. Div. Electronics
via Ghiarola Vecchia, 73
41042 Fiorano (MO) - Italy
tel. 0536/836111 - fax 0536/830901
www.system-group.it
e-mail: info.electronics@system-group.it

Questo prodotto soddisfa i requisiti di protezione **EMC** della direttiva **2004/108/CE.**

1. Introduzione

Il Bridge Ethernet CAN Seriale della famiglia DICO è un prodotto in grado di interfacciare dispositivi CAN o Ethernet a dispositivi seriali (232, 422 o 485). Consente di estendere le moderne architetture di supervisione Ethernet e CAN verso i più consolidati bus seriali.

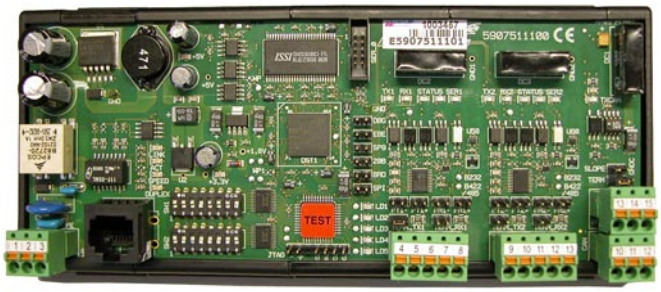


Figura 1.1

2. Specifiche tecniche

- **Microprocessore** DSTni-EX 120MHz x86 compatibile con capacità di indirizzamento a 20 e 24 bits
- 1 **porta ETHERNET** 10/100 Mbps
- 1 **interfaccia full CAN** 2.0A e 2.0B
- 2 **interfacce RS232/RS422/RS485** configurabili a SW
- **Memoria RAM** volatile 2MB DRAM
- **Memoria FLASH** 2MB
- **Memoria EEPROM** seriale 1KB x 2
- **Alimentazione** 24Vdc 0.3A
- **Watch-Dog**
- Possibilità di **programmazione** in linguaggio "C"
- **Temperatura di lavoro:** 0…60 °C
- **Umidità:** UR 85% (senza condensa)
- **Dimensioni:** 195×80×50 mm
- **Montaggio:** su barra DIN EN50035

Mappa di memoria (hw layout con 2MB flash/2MB RAM)					
Addr20	**Size(KB)**	**20-bits**	**24-bits**	**Size(KB)**	**Addr24**
00000	256	On-chip RAM	On-chip RAM	256	000000
40000 7FFFF	256	DRAM	DRAM	1792	040000 1FFFFF
			Sector SA0	64	E00000
			Sector SA1	64	E10000
			…	…	…
			Sector SA7	64	E70000
80000	64	Sector SA8	Sector SA8	64	E80000
90000	64	Sector SA9	Sector SA9	64	E90000
A0000	64	Sector SA10	Sector SA10	64	EA0000
B0000	64	Sector SA11	Sector SA11	64	EB0000
C0000	64	Sector SA12	Sector SA12	64	EC0000
D0000	64	Sector SA13	Sector SA13	64	ED0000
E0000	64	Sector SA14	Sector SA14	64	EE0000
F0000	48	Sector SA15	Sector SA15	64	EF0000
FC000	16	DSTni-EX Bootstrap			
			Sector SA16	64	F00000
			Sector SA17	64	F10000
			…	…	…
			Sector SA30	64	FE0000
			Sector SA31	32	FF0000
			Sector SA32	8	FF8000
			Sector SA33	8	FFA000
			DSTni-EX Bootstrap	16	FFC000

Tabella 2.1

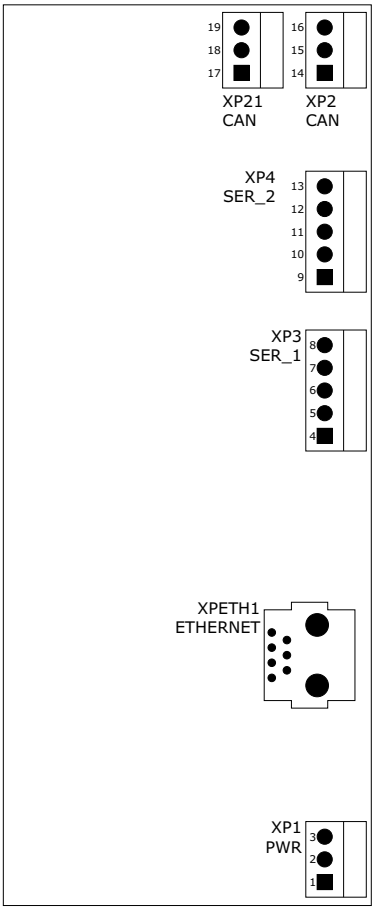


Figura 2.2 Connettori

I parametri di base per l'avvio del modulo e le terminazioni/polarizzazioni dei vari bus sono impostabili via jumper/dip switch. La configurazione del protocollo seriale è impostabile via software e sono disponibili alcuni led di stato per indicare chiaramente il tipo di porta implementato e l'attività in corso.

	STATUS LEDS		
	1	2	3
I/F disabilitata	OFF	OFF	OFF
232	OFF	ON	ON
485	OFF	ON	OFF
422 PP	OFF	OFF	ON
422 MP	ON	OFF	ON

Tabella 2.2

2.1 Jumpers

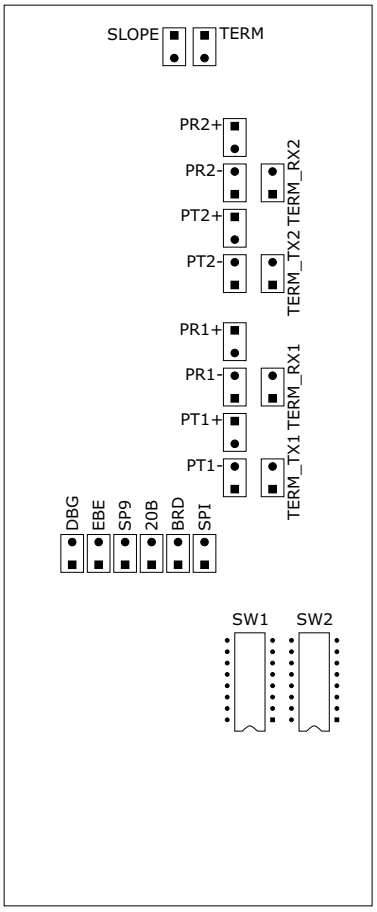


Figura 2.1.1

Configurazione del DSTni-EX Bootloader

DBG = Enable debug message
EBE = Enable boot from Ethernet
SP9 = Set SP 9600
20B = 20 bit mode
BRD = Boot from parallel Flash
SPI = Enable boot from SPI

Configurazione di CAN ed Ethernet

SW1, SW2 = vedi tabella 3.1.1
SLOPE = Slope Control CAN
TERM = Terminazione Linea CAN

Esempi di configurazione

232	Nessun jumper		
485	Polarizzazione (opzionale e solo in un punto della rete)	PT(n)- PT(n)+	PR(n)- PR(n)+
	Terminazione (necessaria per i due nodi estremi della rete)	TERM_TX(n)	TERM_RX(n)
422	Polarizzazione (opzionale e solo in un punto della rete)	PT(n)- PT(n)+	PR(n)- PR(n)+
	Terminazione (necessaria per i due nodi estremi della rete)	TERM_TX(n) Linea TX	TERM_RX(n) Linea RX

Figura 2.1.2

2.2 LEDS

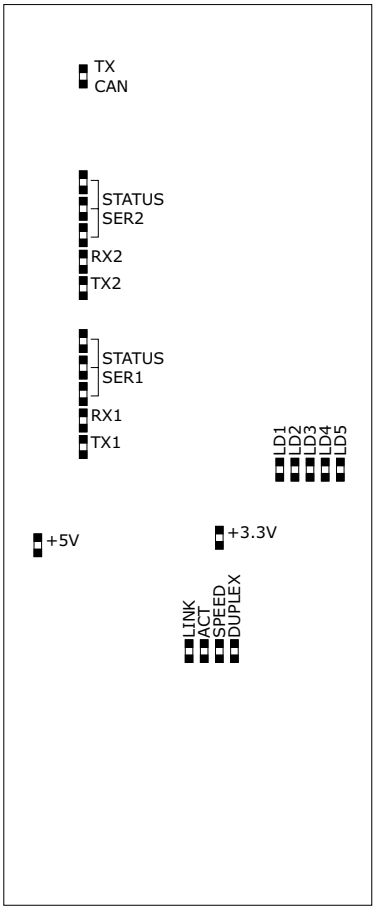


Figura 2.2.1

LINK = Link stabilito
ACT = Segnalazione traffico
SPEED = Velocità connessione 10/100
DUPLEX = Half Duplex / Full Duplex

+5V = Presenza 5V
+3.3V = Presenza 3.3V

LD1 = General purpose
LD2 = General purpose
LD3 = General purpose
LD4 = General purpose
LD5 = General purpose

TX1 = Trasmissione su porta seriale 1
RX1 = Ricezione su porta seriale 1
STATUS SER1 = Tipo di seriale implementata

TX2 = Trasmissione su porta seriale 2
RX2 = Ricezione su porta seriale 2
STATUS SER2 = Tipo di seriale implementata

TX CAN = Attività porta CAN

2.3 CONNESSIONI

XP1 Connettore alimentazione

Pin1 = +24Vdc
Pin2 = GND
Pin3 = PE

XP2 CAN Bus porta 0

Pin14 = CANH
Pin15 = CANL
Pin16 = REF

XP21 CAN Bus porta 0

Pin17 = CANH
Pin18 = CANL
Pin19 = REF

XP3 Seriale SP1

		RS232	RS485	RS422
Pin4	=	TX	B (Data-)	TX-
Pin5	=	RTS	A (Data+)	TX+
Pin6	=	GND	GND	GND
Pin7	=	CTS	NC	RX-
Pin8	=	RXD	NC	RX+

XP4 Seriale SP2

		RS232	RS485	RS422
Pin9	=	TX	B (Data-)	TX-
Pin10	=	RTS	A (Data+)	TX+
Pin11	=	GND	GND	GND
Pin12	=	CTS	NC	RX-
Pin13	=	RXD	NC	RX+

3. MON-EX Firmware

MON-EX è il firmware standard che viene bruciato nella FLASH parallela di Bridge Ethernet CAN Seriale durante la produzione in SYSTEM Electronics. Più precisamente, la FLASH parallela contiene sia la versione a 24-bit che la versione a 20-bit di MON-EX, come evidenziato nelle apposite tabelle contenenti le mappe di memoria.

MON-EX viene lanciato dal bootstrap on-chip e, a sua volta, può fare partire un altro applicativo utente residente ad un indirizzo fisso in FLASH parallela o scaricato in RAM via protocollo BOOTP/TFTP.

SYSTEM Electronics fornisce appositi tools che possono venire usati per comunicare col firmware MON-EX attraverso le porte SP0, CANbus 0 e le porte Ethernet, se attive. Questi tools permettono di conoscere la versione di firmware MON-EX che sta girando, di invalidare un'applicazione esistente, scaricarne una nuova e validarla. Inoltre, sono forniti gli strumenti per utilizzare la seconda EEPROM seriale, l'utilizzo del Real Time Clock, la lettura/scrittura degli I/O e la lettura di zone di memoria.

In caso si voglia inserire in una HMI dedicata il nucleo di questi tools, si possono richiedere a SYSTEM Electronics i dettagli relativi ai protocolli per CANbus e per Ethernet/Seriale.

MON-EX contiene anche un Web Server che permette di interfacciarsi tramite un browser standard (Internet Explorer, Mozilla Firefox, ecc...) per reperire informazioni, impostare IP e Netmask da salvare in memoria non volatile, scaricare applicativi, validarli ed invalidarli.

3.1 Cosa fa MON-EX

La primissima azione di MON-EX è leggere un registro in cui la logica hardware della scheda ha memorizzato la causa di reset della CPU (power-up, watch-dog o altro) e scrivere questa informazione in un area fissa di memoria *(Monitor Exchange Memory)* dove un programma applicativo la possa in seguito reperire.

Quindi, MON-EX legge i dip-switches SW1 e SW2 per sapere come procedere.

SW1_1	SW1_2 SW1_3	SW1_4 SW1_5	SW1_6 SW1_7 SW1_8
CANbus 0 Off = Port used by monitor	**User defined**	**IP Mode** Off Off = Class C *(192.168.1.x) or ARP trick* Off On = E2Prom On Off = DHCP On On = BOOTP	**CANbus baudrate** Off Off Off = 1 Mbps Off Off On = 800 Kbps Off On Off = 500 Kbps Off On On = 250 Kbps On Off Off = 125 Kbps On Off On = 50 Kbps On On Off = 20 Kbps On On On = 10 Kbps
CANbus 0 ON = Port is under TCPtoCAN Gateway			**User defined**

SW2_1	SW2_2 SW2_3 SW2_4 SW2_5 SW2_6 SW2_7 SW2_8
ON = Ignore application program	**Node ID *(CANopen Nid or LSB of Class C IP)*** Off Off Off Off Off Off Off = 0 (No IP; ARP trick enabled) Off Off Off Off Off Off On = 1 (Class C IP = 192.168.1.1) Off Off Off Off Off On Off = 2 (Class C IP = 192.168.1.2) ... ... On On On On On On On = 127 (Class C IP = 192.168.1.127)
	User defined (when CANbus 0 is gateway port)

Tabella 3.1.1

Lo switch più discriminante è, senza dubbio, SW2_1.

Quando è ON, SW2_1 dice a MON-EX di ignorare qualunque programma applicativo eventualmente residente in FLASH parallela e di procedere aprendo un insieme di

porte di comunicazione attraverso cui ricevere comandi dal mondo esterno.

Quando SW2_1 è OFF, MON-EX legge la E2Prom seriale in cerca di patterns magici ed altre informazioni che indichino la presenza di un applicativo utente valido nella FLASH parallela (ad un indirizzo fisso dipendente della modalità di indirizzamento a 20/24 bits) ed eventualmente lo lancia. Per i dettagli relativi agli indirizzi fissi, si faccia riferimento alle tabelle con le opportune mappe di memoria.

È altamente consigliabile che l'applicativo usi i dip-switches relativi a Node ID e baudrate del CANbus nel medesimo modo di MON-EX e che l'applicativo implementi in qualche modo la funzionalità di invalidazione di se stesso (in modo tale da poter essere indotto a suicidarsi e far ripartire MON-EX al successivo reset).

Mappa di memoria di MON-EX-20			
Addr20	**Size (KB)**	**20-bits**	**Usage**
00000	256	On-chip RAM	Interrupt vectors
00400			Reserved
00700			Monitor Exchange Memory
00800			Monitor Data (available to Application)
40000	254	SDRAM	Monitor Free Memory
7F800	2		Not used by Monitor
80000	64	Flash SA8	Application code
90000	64	Flash SA9	
A0000	64	Flash SA10	
B0000	64	Flash SA11	
C0000	64	Flash SA12	Monitor Code MON-EX-20
D0000	64	Flash SA13	
E0000	64	Flash SA14	
F0000	48	Flash SA15	
FC000	16	On-chip ROM	DSTni-EX Bootstrap

Tabella 3.1.2

Mappa di memoria di MON-EX-24 (2 MB Flash / 2MB RAM)			
Addr24	**Size (KB)**	**24-bits**	**Usage**
000000	256	On-chip RAM	Interrupt vectors
000400			Reserved
000700			Monitor Exchange Memory
000800			Monitor Data (available to Application)
040000	1278	SDRAM	Monitor Free Memory
17F800	514		Not used by Monitor
7FFFFF			
E00000	64	Flash SA0	Available to Application
E10000	64	Flash SA1	
...	...	...	
E70000	64	Flash SA7	
E80000	64	Flash SA8-SA27 (1280 KB)	Application Code (started by MON-EX-24)
...	...		
FB0000	64		
FC0000	64	Flash SA31-SA33	Monitor Code MON-EX-24
FE0000	64		
FF0000	32		
FF8000	8		
FFA000	8		
FFC000	16	On-chip ROM	DSTni-EX Bootstrap

Tabella 3.1.3

3.2 Le porte di comunicazione di MON-EX

Quando MON-EX ignora un eventuale programma applicativo, inizia a far lampeggiare Led 5 ed apre alcune porte di comunicazione su cui attendere comandi dall'esterno.

3.2.1 La porta seriale SP0 (RS232)

La **porta seriale SP0** viene aperta a "38400,N81" indipendentemente dal settaggio degli switches.

3.2.2 La porta CAN0

Se lo switch SW1_1 è OFF ed in Node ID è diverso da zero (switches SW2_2 - SW2_8), la porta **CANbus 0** viene aperta con la baudrate impostata dagli switches SW1_6 - SW1_8. Se gli switches SW2_2 - SW2_8 sono tutti OFF, il Node ID viene automaticamente portato da 0 a 1.

L'impostazione di fabbrica per il Node ID è 1; quella per la baudrate è 500Kbps.

Se SW1_1 è ON, la porta CANbus 0 può venire usata come gateway tra TCP e CANbus. Dettagli sul protocollo di gateway sono disponibili per gli interessati.

3.2.3 La porta ETH0

Le porte Ethernet dei dispositivi SYSTEM Electronics hanno MAC del tipo **00-11-63-xx-xx-xx.**

Se riesce a recuperare in qualche modo (dipendentemente dalla posizione di SW1_4 - SW1_5) un IP ed una NetMask, MON-EX apre anche la porta **Ethernet 0.**

Il settaggio di fabbrica per IP_Mode è *Class C*: l'IP è "192.168.1.1" (Node ID, come abbiamo già visto, è impostato ad 1) e NetMask è "255.255.255.0".

Un node ID pari a 0 attiva il cosiddetto "ARP trick", che permette di assegnare un IP temporaneo usando i comandi standard ARP e PING. In dettaglio, si tratta di aggiungere temporaneamente una voce statica alla tabella ARP del proprio PC e, successivamente, di usare il comando PING per assegnare temporaneamente l'IP desiderato alla porta ETH0; ad esempio, se si vuole assegnare temporaneamente l'IP 10.11.12.13 alla porta ETH0 avente MAC address 001163-000100, si devono digitare sul proprio PC i seguenti comandi:

arp -s 10.11.12.13 00-11-63-00-01-00

ping 10.11.12.13

A questo punto, fino al prossimo spegnimento il Bridge Ethernet CAN Seriale diventa raggiungibile (ad esempio, con un browser) attraverso l'IP fissato.

Usando un browser, IP e NetMask per la porta Ethernet 0 possono essere modificati e salvati su E2Prom seriale, in modo che MON-EX possa da lì leggerli in caso IP_Mode venga posto ad *E2Prom*.

Se IP_Mode è impostato a *DHCP*, l'IP e la NetMask per la porta Ethernet 0 può essere ottenuta da un server DHCP; in tal caso SYSTEM Electronics suggerisce che il

server DHCP venga configurato per usare il MAC address come chiave per l'assegnazione di IP e NetMask.

Se IP_Mode è impostato a *BOOTP*, un apposito server deve assegnare IP e NetMask alla porta Ethernet 0 e, quindi, scaricare un applicativo utente nella RAM del Bridge Ethernet CAN Seriale (si veda *Monitor Free Memory* nelle tabelle delle mappe di memoria).

3.3 Le segnalazioni luminose di MON-EX

3.3.1 Led 5

Come già accennato, se MON-EX non deve far ripartire un eventuale programma applicativo, inizia a far lampeggiare Led 5 prima di procedere. Si consiglia, dunque, agli applicatori, di non usare quel led nei programmi, in modo tale che siano chiaramente distinguibili all'esecuzione del Monitor e quella di un applicativo.

3.3.2 Led 4

Se IP_Mode è impostato ad *E2Prom* o a *DHCP* e MON-EX non riesce ad ottenere un IP ed una NetMask validi, esso cessa qualunque attività ed entra in un loop infinito facendo lampeggiare Led 4 alla frequenza di 1Hz.

Se IP_Mode è impostato a *BOOTP* e MON-EX va in timeout sull'attesa di connettersi ad un server apposito, esso ripete continuamente la ricerca del server facendo lampeggiare brevemente Led 4 ad ogni tentativo.

Se IP_Mode è impostato a *Class C* in modo tale da attivare l'*ARP trick*, MON-EX attende l'arrivo del comando PING facendo lampeggiare Led 4 alla frequenza di circa 2Hz.

3.3.3 Led 3

Durante la copiatura da RAM a Flash di un programma applicativo in qualche modo scaricato, MON-EX accende Led 3 per un tempo (breve) che dipende dalla lunghezza dell'applicativo.

3.4 Le E2Prom seriali

Bridge Ethernet CAN Seriale monta 2 chips di memoria E2Prom seriale.

Nel primo di questi chip, protetto contro qualunque tentatvo di scrittura, sono stati memorizzati alcuni dati di fabbrica come il MAC address della porta Ethernet ed altro.

Nel secondo chip, opportunamente scrivibile, possono invece venir memorizzati dati come l'indirizzo IP e la relativa NetMask, i validatori degli eventuali programmi applicativi residenti in Flash parallela nonchè un massimo di 32 bytes utente, cui l'applicativo può assegnare il significato desiderato.

3.5 Configurazione di memoria per gli applicativi

SYSTEM Electronics mette a disposizione degli utilizzatori di Bridge Ethernet CAN Seriale non solo le librerie necessarie a costruire un applicativo, ma anche un esempio di progetto contenente il target MON-EX (completo di sorgenti) ed alcuni altri target che fanno funzionare un piccolo applicativo in varie modalità.

Le note seguenti presuppongono la conoscenza dell'ambiente di sviluppo software Paradigm C/C++ e dei tools messi a disposizione da Lantronix (produttore del chip DSTni-EX).

Senza voler esaurire tutte le possibilità offerte dal chip DSTni-EX e dal suo *bootstrap* interno, nel seguito verranno presentate le varie configurazioni di memoria previste nel file APPL001.CFG; ad ognuna di esse è associato un particolare *style sheet* nel file di progetto APPLICATION.PDL.

Alcune di queste configurazioni tengono conto della eventuale presenza di un Monitor (a 20 o a 24 bits) che, necessariamente, utilizza alcune delle risorse di memoria disponibili; per maggiori dettagli, si faccia riferimento alle tabelle 3.1.2 e 3.1.3 del manuale.

Le risorse impegnate dal Monitor sono attualmente largamente sovrabbondanti rispetto alle sue effettive esigenze (vedi Tabella 3.5.1) e, dunque, non si prevede che sue future versioni possano compromettere il funzionamento di applicativi sviluppati tenendo conto dei vincoli attuali.

Monitor	Dimens. Codice	Flash impegnata	Dimens. Dati	Ram impegnata
20 bits	~ 160 KB	240 KB	~ 162 KB	254 KB
24 bits	~ 160 KB	240 KB	~ 162 KB	254 KB

Tabella 3.5.1

Si noti che lo spazio di ram usato dal Monitor può, ovviamente, essere riutilizzato dall'applicativo utente lanciato dal Monitor stesso.

Si noti infine che Monitor gestisce un'area di scambio con gli applicativi che manda in esecuzione; tale area inizia all'indirizzo fisico 0x00700 ed occupa 256 bytes, così suddivisi:

0x00700 - 0x00701	iRestartReason
0x00702 - 0x00707	szMonitorRelease
0x00708 - 0x00709	iCANbus0Baudrate
0x0070A - 0x0070B	iCANbus0NodeId
0x0070C - 0x007FF	Reserved

Tabella 3.5.2

Tutte le configurazioni descritte nel seguito prevedono dunque che i primi 2KB dello spazio di memoria siano riservati: 1KB servono per la tabella dei vettori di interrupt, 768 bytes per lo stack di PDREM (in caso di debugging) e 256 bytes per l'area di scambio tra Monitor e gli applicativi.

3.5.1 Programma in debugging

Se, nell'ambiente di sviluppo Paradigm, si associa ad un *target* lo *style sheet* "DICO Debug", viene definito il simbolo __PDREMOTE__; in tal caso, il tool di rilocazione (Locator) viene istruito da APPL001.CFG a considerare impegnate le zone di memoria on-chip dedicata a PDREM quando questo viene scaricato al bootstrap attraverso la porta seriale SP0 (da 0x00800 a 0x03FFF e da 0x3F000 a 0x 3FFFF).

Se possibile, si consiglia di usare per l'applicativo la medesima zona di memoria dati che si ha intenzione di usare per il sistema finito.

3.5.2 Programma residente in flash lanciabile dal Bootstrap (esecuzione in flash)

Lo *style sheet* "DICO FLASH stored & executed" definisce il simbolo __IN_FLASH__ che costringe il locator a generare un file in formato HexIntel (FLASH20.HEX o FLASH24.HEX) che dovrà poi essere processato nuovamente da DSTMKBIN32 per ottenere il file FLASH20.SPB (o FLASH24.SPB); questo file deve infine essere scaricato sul target mediante DSTniLoader.

L'indirizzo iniziale del codice deve rispettare le specifiche del bootstrap on-chip, che ricerca l'intestazione di un applicativo in flash agli indirizzi multipli di 64KB (partendo dal più alto e sino all'inizio dell'upper memory).

3.5.3 Programma residente in flash lanciabile da Bootstrap dopo essere stato copiato in ram o lanciabile da Bootstrap dopo averlo ricevuto attraverso la seriale SP0 (esecuzione in ram on-chip)

Lo *style sheet* "DICO FLASH stored, RAM executed" definisce il simbolo __FLASH_TO_RAM__; il file HexIntel generato dal locator (RAM20.HEX o RAM24.HEX) può essere processato da DTMKBIN32 in due diversi modi e generare i files (DTMKBIN32 in due diversi modi e generare i files RAM20.SPB e RAM24.SPB (oppure, eventualmente, i files SERIAL20.SDB e SERIAL24.SDB).

I files .SPB devono essere scaricati sul target mediante DSTniLoader e vengono gestiti da Bootstrap che provvede a copiarli in ram on-chip prima della esecuzione.

I files .SDB devono essere scaricati sul target mediante la porta seriale SP0 e vengono ricevuti da Bootstrap che li copia in ram on-chip e poi li mette in esecuzione.

In entrambi i casi, l'indirizzo di inizio dell'applicativo deve essere obbligatoriamente 0x00800 e l'intero applicativo (codice più dati iniziali) non deve oltrepassare il confine costituito dall'indirizzo 0x3BFFF (max 238KB); questo limite è dovuto all'utilizzo dell'ultima porzione di ram on-chip da parte del Bootstrap e, soprattutto, al fatto che Bootstrap programma i registri di chip select in modo tale da accedere solo alla ram on-chip.

3.5.4 Programma residente in flash lanciabile da Monitor (esecuzione in ram)

Lo *style sheet* "DICO FLASH stored, RAM executed" può anche costringere il locator a generare, oltre al file HexIntel di cui si è detto al punto precedente, anche un file binario RBYMON20.BIN o RBYMON24.BIN. Questo file deve essere scaricato in flash al Monitor attraverso la porta 1100 o mediante il WebServer integrato nel Monitor stesso. Dopo il trasferimento in flash, occorre eseguire l'operazione di validazione dell'applicativo, indicando un indirizzo iniziale non necessariamente uguale a 0x00800.

Monitor provvede a copiare l'applicativo in ram prima dell'esecuzione. *A differenza dei due casi precedenti, avendo Monitor già sistemato opportunamente i registri che governano i chip select delle memorie, l'applicativo non è limitato ad utilizzare la sola ram on-chip,* ma può occupare tutta la memoria lasciata libera da Monitor (fino a 1278KB di copia applicativo per la versione a 24-bits).

3.5.5 Programma residente in flash lanciabile da Monitor (esecuzione in flash)

Lo *style sheet* "DICO FLASH executed by Monitor" definisce il simbolo __FLASH_UNDER_MONITOR__; come nel caso precedente, il file binario generato dal Locator (FBYMON20.BIN o FBYMON24.BIN) deve essere scaricato il flash al Monitor attraverso la porta 1100 o mediante il WebServer integrato nel Monitor stesso. Dopo il trasferimento in flash, occorre eseguire l'operazione di validazione dell'applicativo, indicandone l'indirizzo iniziale (che deve risiedere nelle zone opportunamente messe a disposizione da Monitor). L'esecuzione dell'applicativo avverrà direttamente in flash.

3.5.6 Programma residente su BOOTP e lanciabile da Monitor (esecuzione in RAM)

Lo *style sheet* "DICO BOOTP by Monitor" definisce il simbolo __BOOTP_UNDER_MONITOR; in questo caso, il file binario generato dal locator (BBYMON20.BIN o BBYMON24.BIN) deve essere copiato su BOOTP Server, da cui Monitor lo potrà scaricare se opportunamente configurato tramite i dip-switches presenti sulla scheda (vedere Tabella 3.1.1).

Anche in questo caso, avendo Monitor già sistemato opportunamente i registri che governano i chip select delle memorie, l'applicativo può occupare tutta la memoria lasciata libera da Monitor.

3.5.7 Le classi MEM4STACKS e MEM4BUFFERS

Negli esempi contenuti nel progetto APPLICATION.IDE sono state dichiarate due speciali classi di segmenti per contenere, rispettivamente, il pool di memoria da cui possono venire allocati gli stacks dei tasks ed il pool di buffers per i frames scambiati su Ethernet.

Il fatto che le classi siano distinte permette di dare al Locator direttive per posizionarle dove desiderato. Ad esempio, in un certo applicativo potrebbe essere conveniente posizionare gli stacks nelle ram on-chip (estremamente veloce) ed i buffers di Ethernet in ram esterna (leggermente più lenta); in un altro ancora, potrebbe essere più conveniente portare in ram on-chip porzioni di codice e posizionare tutto il resto in ram esterna.

3.5.8 Utilizzo del floating point

È altresì disponibile la possibilità di utilizzare l'emulazione del coprocessore matematico, utilizzando le librerie floating point presenti in Paradigm C++.

È però ben noto che tali librerie utilizzano una porzione di memoria allocata all'inizio del segmento contenente lo stack. Poichè la customizzazione del sistema operativo DSTniOS operata da SYSTEM Electronics prevede che tutti gli stacks dei vari tasks condividano il medesimo segmento, ne deriva che uno solo dei tasks può eseguire operazioni floating point o che lo sviluppatore dell'applicazione deve provvedere opportuni meccanismi di mutua esclusione.