Corso di Laurea Magistrale in **Ingegneria Informatica**

Elaborato finale per il Corso di Calcolatori Elettronici 2

Prof. Valentina Casola

A.A. 2019/2020

Studente:

Carmine Cesarano MAT. M63/0948

Sommario

1. Specifiche di progetto

Un sistema X è dotato di una periferica seriale e di una periferica parallela. Il sistema trasmette un messaggio di 32 caratteri (byte) sulla periferica seriale e ottiene l'eco del messaggio sulla periferica parallela. L'ipotesi di funzionamento scelta è la seguente:

-> Il messaggio è trasmesso interamente dalla seriale e successivamente è ricevuto mediante il meccanismo delle interruzioni sulla periferica parallela. Non può essere inviato un altro messaggio sulla seriale se non è stato ricevuto prima l'eco sulla parallela.

Si illustrino:

- l'architettura complessiva del sistema;
- il collegamento tra i dispostivi;
- i protocolli;
- il software e la memoria con riferimento a dati e programmi in essa allocati

2. Soluzione

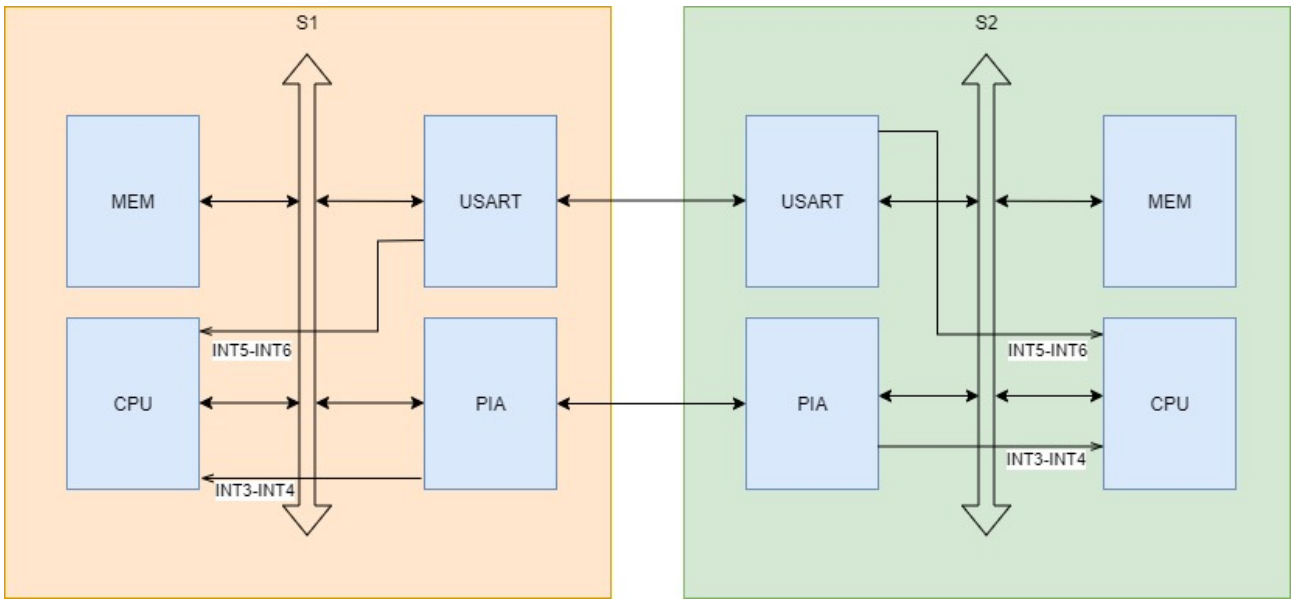
Prima di esporre la soluzione è necessario precisare alcune assunzioni fatte sul progetto: il messaggio da trasmettere e per il quale si deve ricevere l'eco prima di abilitare una nuova trasmissione, è *l'intero messaggio di 32 caratteri*.

Dunque, i due sistemi vengono sincronizzati ogni 32 caratteri trasferiti o ricevuti:

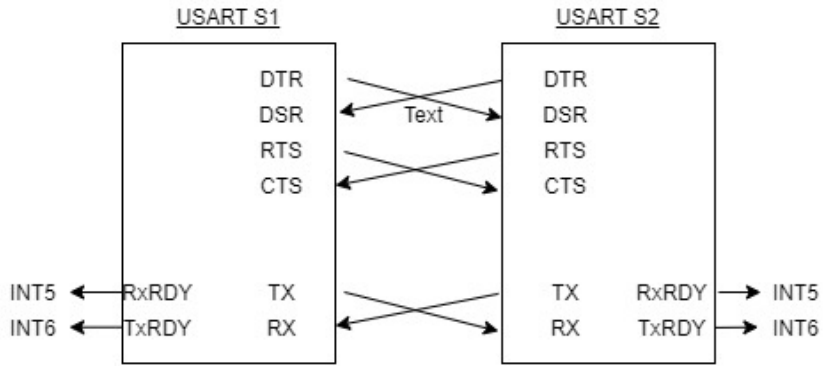
1. S1 trasmette 32 caratteri (un messaggio) tramite interfaccia seriale;
2. S2 riceve 32 caratteri (un messaggio) tramite interfaccia seriale e li salva in memoria;
3. Al termine della ricezione, S2 inizia a trasmettere i 32 caratteri salvati (ECO) tramite interfaccia parallela;
4. S1 riceve tutti e 32 caratteri di ECO sull'interfaccia parallela, e solo al termine della ricezione può abilitare la trasmissione di un nuovo messaggio di 32 caratteri (sulla seriale).

Per cui globalmente il sistema di comunicazione S1-S2 implementa una comunicazione *sincrona* perchè S1 sa se il messaggio è arrivato a destinazione, e in ottica locale, *bloccante*, in quanto il controllo è restituito ad S1 solo quando la comunicazione è stata completata.

2.1. Architettura del sistema

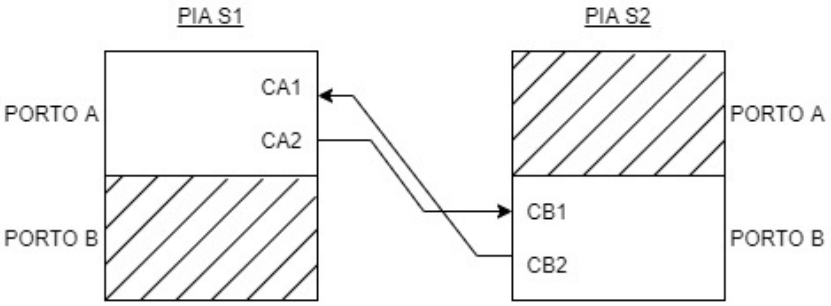


Per quanto riguarda la **comunicazione seriale**, tramite la USART viene utilizzato un tipo di handshaking semplificato detto FULL NULL-MODEM, che collega direttamente i due terminali come segue:



In questo modo se i sistemi sono inizializzati ponendo DTR e RTS ad 1, l'handshaking è stabilito in partenza. La USART del primo sistema è configurata per trasmettere mentre quella del secondo per ricevere, ed inoltre è stata scelta una modalità di trasmissione asincrona per le due periferiche.

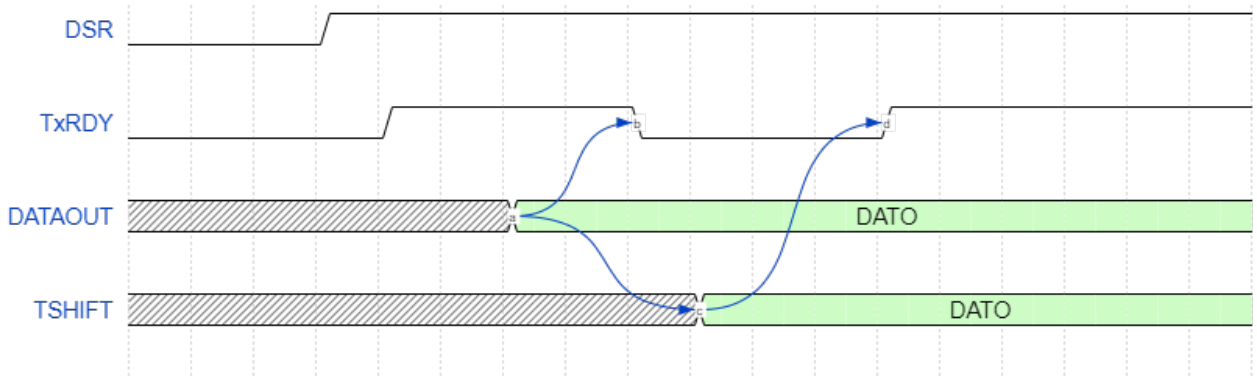
Per quanto riguarda la **comunicazione parallela,** tramite la PIA, viene utilizzata la modalità 'handshake', e vengono collegati i due terminali come segue:



2.2. Protocolli

2.2.1. Tempificazione USART

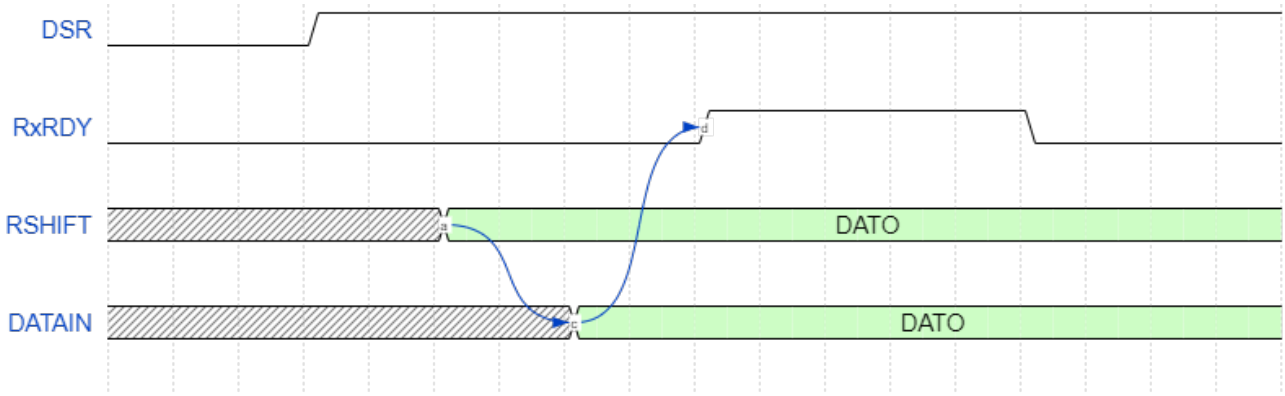
Il sistema S1 si serve della USART per trasmettere, in un loop, 32 caratteri al sistema S2. Di seguito la tempificazione delle operazioni di **scrittura**:



1. Controlla che DSR sia alto: in partenza, DTR e RTS della periferica gemella sono attivi (configurati tramite CNTRL), dunque per la configurazione, anche DSR sarà attivo. Il bit 0 del registro di stato (TxRDY) è alto.
2. Controlla che il bit 0 sia alto, quindi che la periferica sia pronta a trasmettere;
3. Scrive il carattere in DATAOUT
4. Il bit 0 del registro di stato si abbassa
5. Alla fine della trasmissione del carattere, il bit 0 si alza e viene attivato il segnale di interruzione TxRDY
6. TxRDY è collegato a RxRDY del sistema 2 (che può iniziare la lettura)

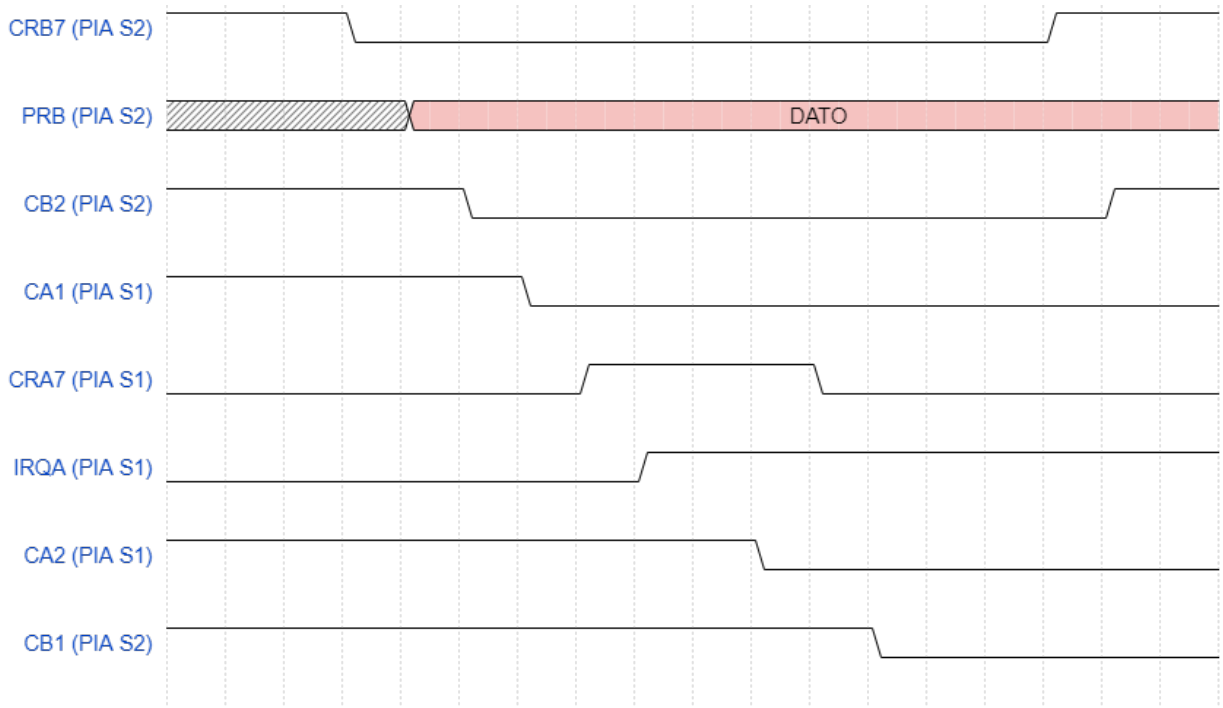
Il sistema S2, tramite la USART deve leggere i 32 caratteri. Di seguito la tempificazione della **lettura**:

1. Il segnale RxRDY scatena un'interruzione sul sistema S2, e la ISR che la gestisce dovrà leggere il dato su DATAIN della USART.
2. Quando viene letto il dato, il bit 1 del registro di stato, che corrisponde a RxRDY, torna a zero



2.2.2. Tempificazione PIA

Il sistema S2 deve trasmettere l'eco del messaggio tramite periferica PIA, mentre S1 riceve tramite PIA. Di seguito il diagramma temporale della **scrittura**:



1. L'ipotesi di partenza è CRB7=0. Il sistema S2 attende che CRB7=1 per scrivere un dato.
2. Il sistema S2 effettua una lettura fittizia per abbassare CRB7 ed iniziare la scrittura
3. Il sistema S2 scrive un dato sul porto B (PRB) della PIA trasmittente (MOVE (A0)+,PIADB);
4. L'operazione di scrittura sul porto B fa abbassare CB2 del sistema S2, quindi automaticamente, per come abbiamo collegato, CA1 del sistema S1 si abbasserà;
5. Il flag CRA7 del sistema S1 si alza, indicando che può iniziare l'operazione di lettura;
6. La transizione 1->0 di CA1 fa scattare l'interruzione IRQA del sistema S1, gestita dalla relativa ISR;
7. La ISR sul sistema S1 provvederà a leggere il dato sul porto A della PIA.
8. Il flag CRA7 e il segnale CA2 del sistema S1 si abbassano
9. Il segnale CB1 del sistema S2 si abbassa
10. Il segnale CRB7 del sistema S2 si alza

2.3. Mappa della memoria

Di seguito è riportata la mappa della memoria RAM nei due sistemi:

	Sistema S1
8000	01 02 03 04 05 06 07 08
8008	09 0A 0B 0C 0D 0E 0F 10
8010	11 12 13 14 15 16 17 18
8018	19 1A 1B 1C 1D 1E 1F 20
8020	20 1F 1E 1D 1C 1B 1A 19
8028	18 17 16 15 14 13 12 11
8030	10 0F 0E 0D 0C 0B 0A 09
8038	08 07 06 05 04 03 02 01
8040-8078	**64byte per ECO1-ECO2**
8080	**DIM, MSGTOT, MSGINV, CONT, FLAG**
....	
8200	**MAIN**
.....	
8700	**INT3** (ISR PIA-A)
....	
8800	**INT6** (ISR USART)

	Sistema S2
8000	**MSG, DIM, COUNT**
....	
8200	**MAIN**
....	
8700	**INT5** (ISR USART)

Di seguito è riportata la mappatura dei *vettori delle interruzioni* nella memoria ROM, relativa ai due sistemi:

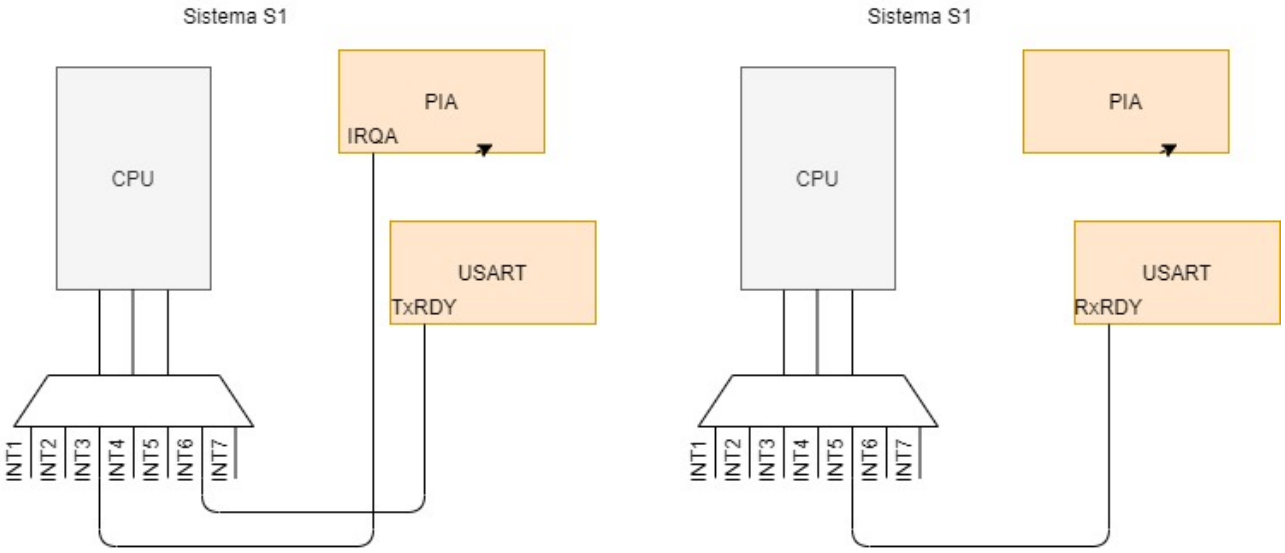
	Sistema S1
006C	00008700
0070	
0074	
0078	00008800

	Sistema S2
006C	
0070	
0074	00008700
0078	

Tale mappatura dipende dalla scelta delle linee di interruzione dei diversi dispositivi. Sia nel sistema S1 che nel sistema S2, i dispositivi PIA ed USART sono collegati al processore come segue:

- USART: linee di interruzione INT5-INT6
- PIA: linee di interruzioni INT3-INT4

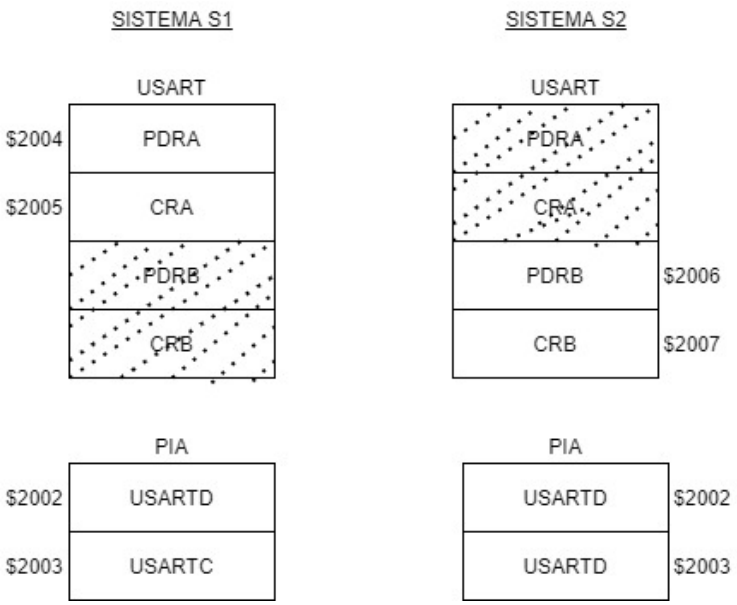
In particolare:



Utilizzando la tecnica *autovettorizzata* con il MC68K, i dispositivi vengono identificati automaticamente ed il loro vector number calcolato a partire dalla linea di interruzione. Ad esempio:

-> La linea IRQA della PIA del sistema S1 è collegata alla linea di interruzione INT3 dunque calcolo il vector number come (IPL+24 = 3+27 = 27), moltiplico per 4 e converto in esadecimale, ottenendo $6C. Vuol dire che all'indirizzo $6C in ROM deve essere scritto l'indirizzo della ISR di gestione della PIA in ricezione.

Dato che sia la USART che la PIA sono dispositivi *memory mapped*, i registri interni delle interfacce sono mappati in memoria centrale. Di seguito è rappresentata la mappatura in memoria dei dispositivi:

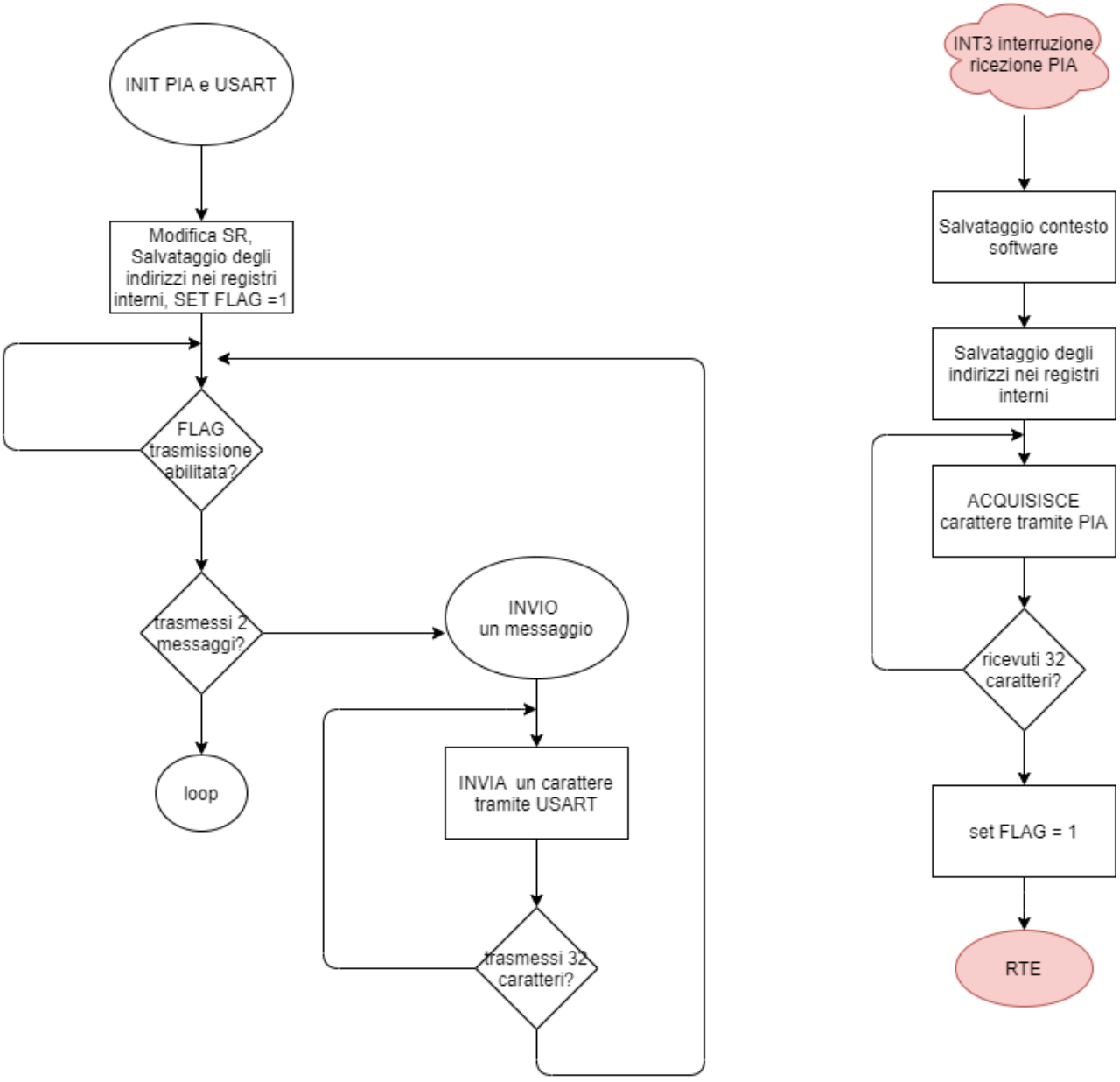


3. Implementazione

3.1. Documentazione del driver S1

Nel sistema S1 viene mantenuto in memoria un flag che abilita la trasmissione di un messaggio tramite la USART. La trasmissione del primo messaggio è abilitata di default (FLAG=1), le successive sono abilitate solo se è stato ricevuto completamente l'eco del messaggio sulla PIA. Per testare questo funzionamento sono stati memorizzati in memoria 2 messaggi di 32 byte.

Quando viene abilitata la trasmissione, la USART trasmette tutti e 32 caratteri del messaggio al sistema S2.

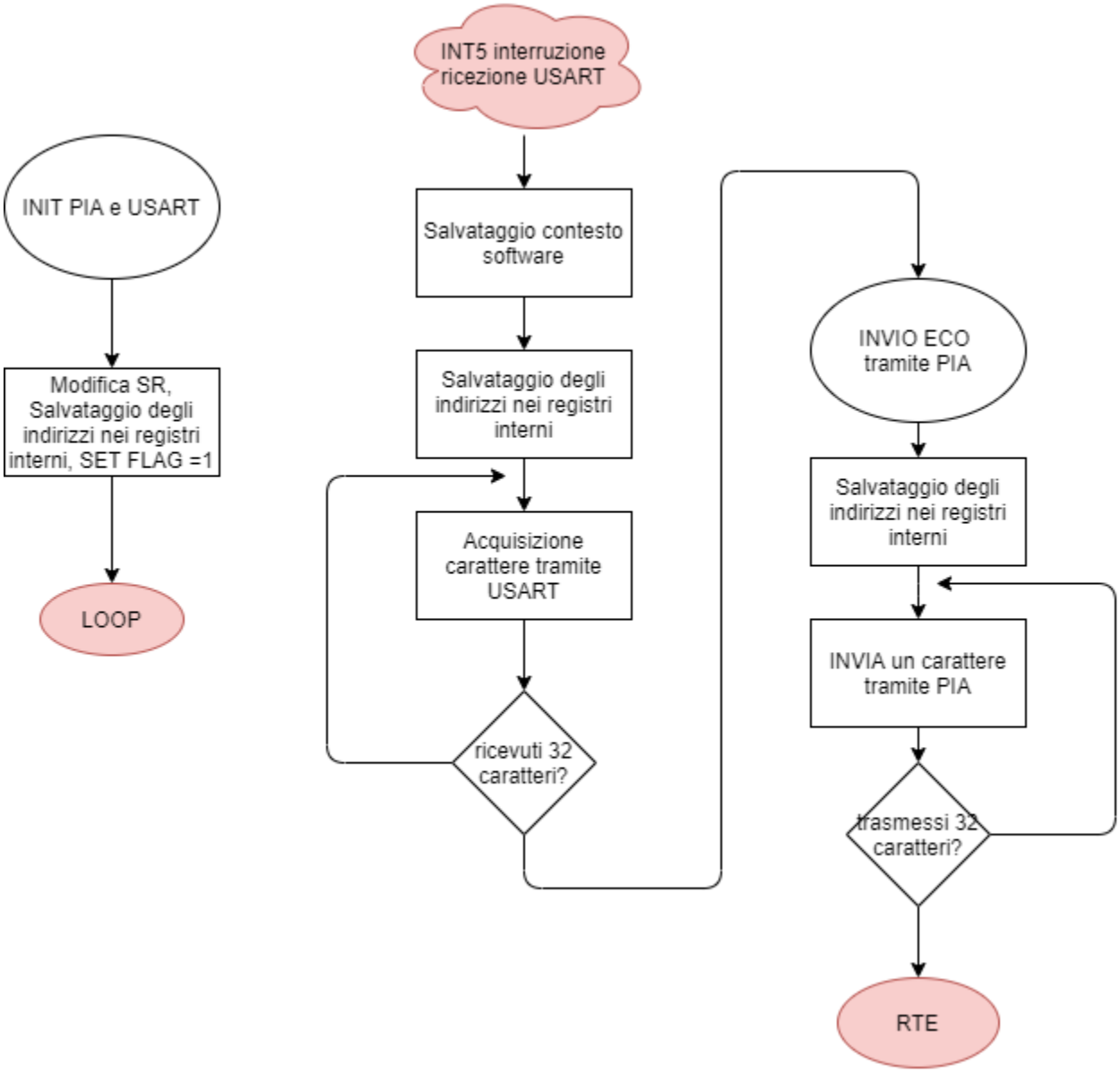


Quando il sistema S1 riceve un'interruzione sulla linea INT3, viene gestita dalla ISR il cui flusso è schematizzato a destra. Dopo aver salvato il contesto software sullo stack e gli indirizzi utili nei registri interni avviene l'acquisizione dei 32 caratteri tramite la periferica PIA. Questi caratteri corrispondono all'eco del messaggio inviato precedentemente dallo stesso S1, e ricevuto da S2.

3.2. Documentazione del driver S2

Il sistema S2 viene abilitato su interruzione INT5 alla ricezione di un carattere tramite la porta seriale USART. Dopo aver salvato il contesto software e gli indirizzi utili nei registri interni avviene l'acquisizione dei 32 caratteri tramite la periferica USART.

Questi caratteri vengono salvati nella memoria del sistema S2, e al termine dell'acquisizione viene abilitata la trasmissione degli stessi tramite la periferica PIA verso il sistema S2.



3.3. Codice Assembly

	SISTEMA S1
AREA DATI	ORG $8000 MSG1 DC.B 1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30,31,32 MSG2 DC.B 32,31,30,29,28,27,26,25,24,23,22,21,20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1 ECO1 DS.B 32 ECO2 DS.B 32 DIM DC.B 32 ;dimensione del singolo messaggio MSGTOT DC.B 2 ;totale dei messaggi (di 32byte) da inviare MSGINV DC.B 0 ;contatore messaggi (di 32byte) inviati CONT DC.B 0 ;contatore dei caratteri di eco ricevuti nell'ambito del messaggio. Azzerato ogni 32 ECOCHAR DC.B 0 ;offset dell'area di memoria ECO1 per salvare i caratteri ricevuti FLAG DC.B 1 ;flag che indica la ricezione di tutti i caratteri ECO e abilita la trasmissione

MAIN	ORG $8200
	USARTD EQU $2002 ;registro dato USART
	USARTC EQU $2003 ;registro di controllo USART
	PIADA EQU $2004 ;indirizzo di PIA-A dato, usato in input
	PIACA EQU $2005 ;indirizzo di PIA-A stato/controllo
	
	MAIN
	JSR INITUSART ;inizializzazione USART
	JSR INITPIA ;inizializzazione PIA
	
	MOVE.W SR,D0 ;legge il registro di stato
	ANDI.W #$D8FF,D0 ;maschera per reg stato (stato utente, int abilitati)
	MOVE.W D0,SR ;pone liv int a 000
	
	* salvataggio indirizzi utili nei registri interni
	MOVEA.L #USARTD,A1
	MOVEA.L #USARTC,A2
	MOVEA.L #MSG1,A0
	MOVE.B DIM,D3 ;dimensione singolo messaggio
	MOVE.B MSGTOT,D5 ;messaggi totale da trasmettere
	MOVE.B MSGINV,D6 ;contatore messaggi correntemente inviati
	
	CLR D1 ;appoggio
	CLR D2 ;contatore caratteri trasmessi
	
	CHECKDSR
	MOVE.B (A2),D0 ;Controlla se è attivato il segnale DSR della USART ed in caso affermativo,
	ANDI.B #$80,D0 ;trasmette altrimenti attende. Poichè la configurazione prevede che DTR=1
	BEQ CHECKDSR
	
	STOP
	MOVE.B FLAG,D4 ;lettura del flag che abilita la trasmissione.
	CMPI #1,D4
	BNE STOP ;Attendiamo fin quando l'ECO non è stato completamente ricevuto
	
	ANDI #$00,D4 ;Se l'ECO è stato ricevuto completamente setto il flag a 0
	MOVE.B D4,FLAG
	
	CMP D5,D6 ;Se ho trasmesso 2 messaggi mi metto in loop, altrimenti prosegue con la trasmissione
	BEQ LOOP

```
                CHECKTxRDY
                        MOVE.B  (A2),D0         ;Controlla se è attivo TxRDY, (la seriale è pronta a trasmettere)
                        ANDI.B  #$01,D0
                        BEQ     CHECKTxRDY

                        MOVE.B  (A0)+,D1
                        MOVE.B  D1,(A1)         ;invio il carattere sulla porta DATO della seriale
                        ADD.B   #1,D2           ;incremento il contatore dei caratteri trasmessi

                        CMP     D2,D3
                        BNE     CHECKTxRDY


                        CLR     D2      ;azzero il contatore dei caratteri trasmessi (il cont è riferito al singolo messaggio)

                        ADDI.B  #$01,D6
                        MOVE.B  D6,MSGINV       ;incremento il contatore dei messaggi inviati
                        JMP     STOP            ;mi metto in attesa che il flag abiliti la prossima trasmissione

LOOP    JMP     LOOP
```

```
SUBR            INITUSART
                        MOVE.B  #$5D,USARTC     ; asincrona, 8 bit di informazione, bit di parità dispari e 2 bit di stop.  (MODE)
                        MOVE.B  #$23,USARTC     ;abilita il trasmettitore e attiva i segnali di handshaking. (CONTROL)
                        RTS
```

```
SUBR            INITPIA
                        MOVE.B  #0,PIACA        ;il prossimo accesso a PIADA (indirizzo pari) seleziona il registro DRA
                        MOVE.B  #$00,PIADA              ;accede a DRA e pone DRA=0: le linee di A sono linee di input
                        MOVE.B  #%00100101,PIACA        ;imposta il registro di controllo ponendo IRQA1=1 e IRQA2=1
                        RTS
```

```
ISR1
(PIA_RX)        ORG $8700

INT3
                        MOVE.L  A1,-(A7)        ;salvataggio registri
                        MOVE.L  A0,-(A7)
                        MOVE.L  D0,-(A7)
                        MOVE.L  D1,-(A7)
                        MOVE.L  D2,-(A7)
                        MOVE.L  D3,-(A7)

                        * salvataggio indirizzi utili nei registri interni
                        MOVEA.L  #PIADA,A1
                        MOVEA.L  #ECO1,A0       ;indirizzo di partenza area salvataggio ECO
                        MOVE.B   ECOCHAR,D0     ;offset per salvare l'eco ricevuto nell'area di memoria ECO1
                        MOVE.B   DIM,D1         ;dim del singolo messaggio (32)
                        MOVE.B   FLAG,D2        ;flag che indica la completa ricezione dell'eco e abilita la trasmissione
                        MOVE.B   CONT,D3        ;contatore dei caratteri di eco ricevuti (si azzera ogni 32)

                        MOVE.B  (A1),(A0,D0)    ;acquisisce il carattere e lo trasferisce in memoria

                        ADD.B   #1,D0
                        ADD.B   #1,D3
                        MOVE.B  D0,ECOCHAR      ;incrementa l'offset
                        MOVE.B  D3,CONT         ;incrementa il contatore dei char ricevuti

                        CMP     D3,D1           ;se abbiamo ricevuto 32 char, setta il flag ad 1
                        BNE     FINE

                        MOVE.B  #$00,CONT       ;azzeramento contatore di messaggio
                        ORI     #$01,D2         ;set flag a 1
                        MOVE.B  D2,FLAG

FINE
```

```
                MOVE.L  (A7)+,D3        ;ripristino registri
                MOVE.L  (A7)+,D2
                MOVE.L  (A7)+,D1
                MOVE.L  (A7)+,D0
                MOVE.L  (A7)+,A0
                MOVE.L  (A7)+,A1

                RTE
ISR2 (USART TX) ORG $8800
INT6
                NOP
                RTE

END             MAIN
```

	SISTEMA S2
AREA DATI	ORG $8000 MSG DS.B 32 *area di memoria appoggio per il messaggio da inoltrare ad S1 DIM DC.B 32 *dimensione del singolo messaggio COUNT DC.B 0 *contatore dei caratteri ricevuti nell'ambito del messaggio (si azzera ogni 32)
MAIN	ORG $8200 USARTD EQU $2002 ;indirizzo dato USARTC EQU $2003 ;indirizzo stato/controllo PIADB EQU $2006 ;indirizzo di PIA-B dato, usato in output PIACB EQU $2007 ;indirizzo di PIA-B controllo MAIN JSR INITUSART ;inizializzazione USART JSR INITPIA ;inizializzazione PIA MOVE.W SR,D0 ;legge il registro di stato ANDI.W #$D8FF,D0 ;maschera per reg stato (stato utente, int abilitati) MOVE.W D0,SR ;pone liv int a 000 MOVEA.L #USARTC,A2 ;indirizzo registro controllo/stato CHECKDSR MOVE.B (A2),D0 ;Controlla se è attivato il segnale DSR della USART ed in caso affermativo trasmette, ANDI.B #$80,D0 ;altrimenti attende. Poichè la configurazione prevede che DTR=1 BEQ CHECKDSR LOOP JMP LOOP ;ciclo caldo dove il processore attende interrupt
SUBR	INITUSART MOVE.B #$5D,USARTC ; asincrona, 8 bit di informazione bit di parità dispari e 2 bit di stop. (MODE) MOVE.B #$36,USARTC ;abilita ricevitore, cancella flags (CONTROL) RTS
SUBR	INITPIA MOVE.B #0,PIACB ;seleziona il registro direzione di PIA porto B MOVE.B #$FF,PIADB ;accede a DRB e pone DRA=1 : le linee di B sono linee di output MOVE.B #%00100100,PIACB RTS
ISR1 (USART RX)	ORG $8700 INT5 MOVE.L A3,-(A7) ;salvataggio registri MOVE.L A2,-(A7) MOVE.L A1,-(A7) MOVE.L A0,-(A7) MOVE.L D0,-(A7) MOVE.L D1,-(A7)

```
                MOVE.L  D2,-(A7)
                MOVE.L  D3,-(A7)

                MOVEA.L  #USARTD,A1             ;registro dato USART
                MOVEA.L  #MSG,A0                ;indirizzo area di salvataggio
                MOVE.B   DIM,D1                 ;dimensione del messaggio
                MOVE.B   COUNT,D0               ;contatore caratteri ricevuti

RICEZ
                MOVE.B  (A1),(A0,D0)    ;riceve un carattere e lo memorizza
                ADD.B   #1,D0           ;incremento il contatore dei caratteri ricevuti
                MOVE.B  D0,COUNT

                CMP     D0,D1           ;se abbiamo ricevuto 32 caratteri procede con la trasmissione parallela dell'eco
                BEQ     INVIOECO

FINE
                MOVE.L  (A7)+,D3        ;ripristino registri
                MOVE.L  (A7)+,D2
                MOVE.L  (A7)+,D1
                MOVE.L  (A7)+,D0
                MOVE.L  (A7)+,A0
                MOVE.L  (A7)+,A1
                MOVE.L  (A7)+,A2
                MOVE.L  (A7)+,A3

                RTE
```

SUBR (invio PIA)

```
INVIOECO
                MOVE.B   #$00,COUNT     ;azzero il contatore degli elementi ricevuti

                MOVEA.L  #PIACB,A2      ;indirizzo registro di controllo CRB
                MOVEA.L  #PIADB,A3      ;indirizzo registro PRB

                CLR     D2              ;appoggio
                CLR     D3              ;contatore locale elementi trasmessi

INVIO
                MOVE.B  (A3),D2         ;lettura fittizia da PRB => serve per azzerare CRB7 dopo il primo carattere
                MOVE.B  (A0)+,D2        ;carattere corrente da trasferire in D2;
                MOVE.B  D2,(A3)         ;dato su bus di PIA porto B: dopo la scrittura di PRB, CB2 si abbassa

                ADD.B   #1,D3           ;incremento contatore elementi trasmessi

ciclo
                MOVE.B  (A2),D2         ;In attesa di DATA ACKNOWLEDGE
                ANDI.B  #$80,D2         ;aspetta che CRB7 divenga 1
                BEQ     ciclo

                CMP     D3,D1           ;controlla se ho trasmesso tutti e 32 caratteri
                BNE     INVIO

                JMP FINE

                END     MAIN
```

3.4. Simulazione in ASIM

Di seguito è mostrata la configurazione ASIM per la simulazione del codice prodotto.

```
          Configuration name: config.cfg

          CHIP Name: MEM1
Type: MMU/BUS.   Identif: 01.          BUS: 0000.
Addres 1: 00008000.          Address 2: 00000000.
Com1: 0000.  Com2: 0010.  Com3: 0008.  Com4: 0000.

          CHIP Name: M68000
Type: CPU.          Identif: 02.          BUS: 0001.
Addres 1: 00009000.          Address 2: 00009200.
Com1: 0000.  Com2: 0000.  Com3: 0000.  Com4: 0000.

          CHIP Name: I8251USART
Type: Device.       Identif: 03.          BUS: 0001.
Addres 1: 00002002.          Address 2: 00002003.
Com1: 0002.  Com2: 0005.  Com3: 0006.  Com4: 0007.

          CHIP Name: M6821PIA
Type: Device.       Identif: 04.          BUS: 0001.
Addres 1: 00002004.          Address 2: 00002007.
Com1: 0002.  Com2: 0003.  Com3: 0004.  Com4: 0208.

          CHIP Name: MEM2
Type: MMU/BUS.   Identif: 05.          BUS: 0000.
Addres 1: 00008000.          Address 2: 00000000.
Com1: 0000.  Com2: 0010.  Com3: 0008.  Com4: 0000.

          CHIP Name: M68000
Type: CPU.          Identif: 06.          BUS: 0005.
Addres 1: 00009000.          Address 2: 00009200.
Com1: 0000.  Com2: 0000.  Com3: 0000.  Com4: 0000.

          CHIP Name: I8251USART
Type: Device.       Identif: 07.          BUS: 0005.
Addres 1: 00002002.          Address 2: 00002003.
Com1: 0006.  Com2: 0005.  Com3: 0006.  Com4: 0003.

          CHIP Name: M6821PIA
Type: Device.       Identif: 08.          BUS: 0005.
Addres 1: 00002004.          Address 2: 00002008.
Com1: 0006.  Com2: 0003.  Com3: 0004.  Com4: 0204.
```