



Traccia

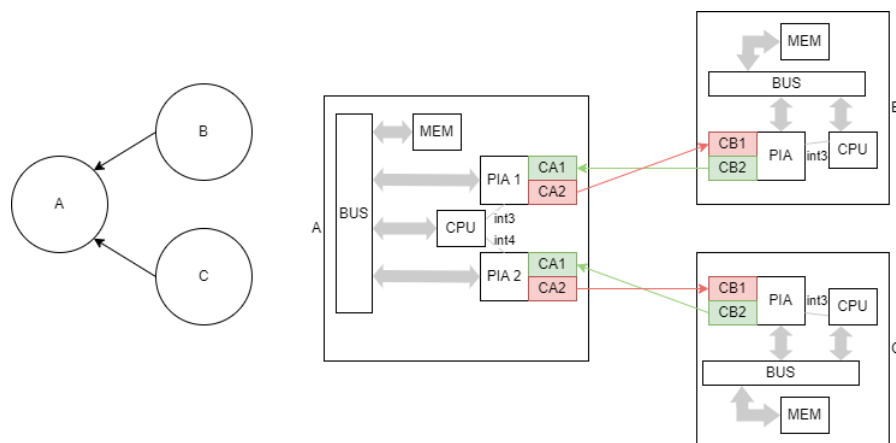
Un sistema è composto da 3 unità, A, B e C. A è collegato in ricezione a B e C mediante due periferiche parallele P1 e P2. B e C sono programmati per inviare continuamente messaggi da N caratteri ad A, ma la ricezione deve essere gestita in modo tale da assicurare che **in ogni istante il numero di messaggi ricevuti da B sia sempre superiore al numero di messaggi ricevuti da C**. Si progetti l'unità A specificando:

- l'architettura complessiva;
- i protocolli;
- la mappa della memoria;
- una descrizione di alto livello del programma implementato;
- l'implementazione in linguaggio assembly Motorola 68000.

Si descrivano elencandole le modifiche hardware e software necessarie nel caso in cui si utilizzi un DMA per la ricezione dei messaggi da B e C, proponendo eventualmente delle modifiche alla logica per meglio adattarsi alla natura delle periferiche coinvolte.

Architettura

Il nodo A riceve da B e C tramite due periferiche, che saranno collegate ai porti B delle PIA nei nodi B e C. In particolare l'architettura è la seguente:



Visto che i messaggi che devo ricevere da B devono essere in numero maggiore rispetto a quelli che ricevo da C, si pone ad un livello di interruzione minore il nodo C, come indicato in figura.

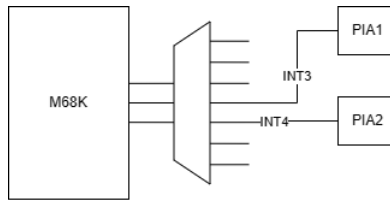
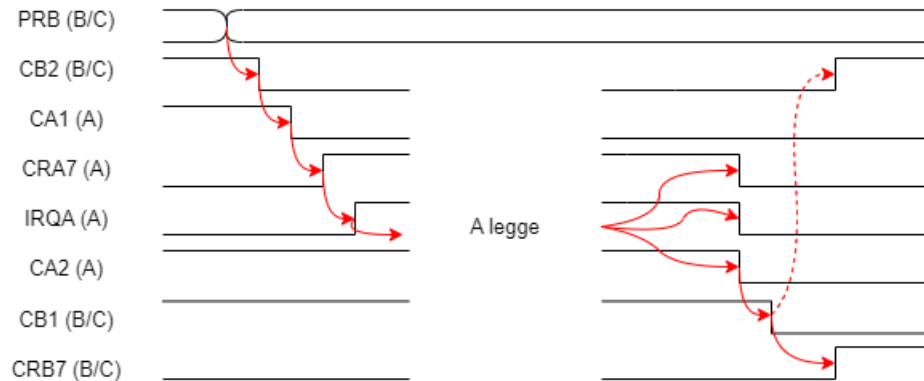


Figura 1: Architettura delle interruzioni nel nodo A

Protocolli

La comunicazione tra A e i nodi trasmettenti B e C è regolata tramite il protocollo di handshaking 100 della PIA, descritto tramite il diagramma temporale seguente.



Tale diagramma vale sia per la comunicazione con B che con C. I due nodi infatti sfruttano la stessa periferica configurata allo stesso modo.

Mappa della memoria

La memoria RAM è indirizzata a word, quindi ogni locazione pari conterrà una word. Ponendo tutte le variabili pari ad un byte, le variabili occuperanno locazioni contigue.

La memoria ROM invece contiene alle posizioni dedicate alle interruzioni, gli indirizzi iniziali delle ISR che si vanno ad implementare.

MEMORIA RAM		MEMORIA ROM	
\$8000	TURN0 FASE	\$6C	00 00 85 00
\$8002	COUNT_CAR COUNT_MESS	\$70	00 00 87 00
\$8004	MESSAGGIO MESSAGGIO		
\$8006	MESSAGGIO MESSAGGIO		
	...		
\$8200	MAIN		
	...		
\$8500	ISR_B		
	...		
\$8700	ISR_C		
	...		

Pseudocodice

Per fare sì che il nodo A riceva sempre un numero di messaggi maggiore da B rispetto che da C, non basta gestire una sorta di alternanza tra i nodi B e C, ma è necessario che B, al primo invio, invii due messaggi consecutivi ad A.

Se avessimo che il nodo B manda il suo messaggio sempre per primo, senza che nella prima fase ne abbia mandati 2 consecutivi, si avrebbe che in un certo istante i messaggi di B e C siano in ugual numero. Per risolvere si è scelto di ipotizzare che il nodo A svolga 2 fasi:

- **FASE 1:** B invia due messaggi consecutivi ad A e poi provvede a risvegliare C;
- **FASE 2:** B e C si alternano, facendo andare prima C dopo la fase 1.

Lo pseudocodice è quindi riportato di seguito.

```
1  MAIN(){
2      inizializza PIA 1 in ricezione;
3      inizializza PIA 2 in ricezione;
4      Enable stato utente;
5      Enable interruzioni;
6      while(true);           //attendo l'interruzione
7  }
8
9  /*
10 FLAG UTILIZZATI:
11 - FASE: flag usato per stabilire la fase in cui si trova il nodo A, che vale 0 se mi
    trovo in fase 1, e vale 1 se mi trovo in fase 2
12 - TURNO: flag che serve per fare l'alternanza in fase 2, vale 0 se e' il turno di C e
    vale 1 se il turno e' di B.
13 */
14 ISR_B(){
15     if(FASE==0){
16         leggi da PIA1;
17         count_car ++;
18         if(count_car == N){
19             count_car = 0;
20             count_mess ++;
21             if(count_mess == 2){
22                 FASE = 1;
23                 TURNO = 1;
24                 if(CRA7(C) == 1){
25                     leggi da PIA2;
26                     count_car ++;
27                 }
28             }
29         }
30     }
31     else if(TURNO == 1){    //Se non e' il turno di B, B non puo' mandare
32         leggi da PIA1;
33         count_car ++;
34         if(count_car == N){ //termina l'invio di un messaggio da B
35             count_car = 0;
36             count_mess ++;
37             TURNO = 0;      //riattivazione di C
```

```

38         if(count_mess == M){
39             disattiva PIA1;
40         }
41         if(CRA7(B) == 1){
42             leggi da PIA2;
43             count_car ++;
44         }
45     }
46 }
47 return from ISR;
48 }
49
50 ISR_C(){
51     if(TURNO == 0){
52         leggi da PIA2;
53         count_car ++;
54         if(count_car == N){
55             count_car = 0;
56             count_mess ++;
57             TURNO = 1;          //riattivazione di B
58             if(count_mess == M){
59                 disattivazione PIA2;
60             }
61             if(CRA7(B) == 1){
62                 leggi da PIA1;
63                 count_car ++;
64             }
65         }
66     }
67     return from ISR;
68 }

```

Scenari

Vediamo che cosa accade nei differenti casi di interruzioni:

B invia prima di C, e C interrompe dopo

```

B  x x x x
C      # x x x x

```

Quando si verifica questa situazione, non ho alcun problema visto che B ha terminato e ha quindi risvegliato il nodo C, che può tranquillamente trasmettere il suo messaggio.

C vuole trasmettere mentre lo sta facendo B

```

B  x x x x
C      # ... x x x x

```

Grazie al flag, che verrà riattivato solo quando B ha finito, C attende la ricezione completa del messaggio da B e poi invia il suo. Anche qui non ho alcun problema.

C interrompe prima del nodo B

In tale occasione devo distinguere due casi: se si tratta di un messaggio di C che arriva quando B ha terminato il suo inoltro non ci sarà alcun problema, ma altrimenti C deve comunque attendere il suo turno, se B è stato il primo ad attivarsi.

```
B      x x x x
C      # ..... x x x x
```

C interrompe in fase 1

```
B      x x x x      x x x x
C      # ..... x x x x
```

Se sono in fase 1, C deve attendere due differenti messaggi da B, al termine dei quali potrà inviare il suo al nodo A.

Programma assembly

```
1  * ----- AREA DATI -----
2      ORG      $8000
3  N      EQU      4          *dimensione di un messaggio
4  M      EQU      10        *numero totale di messaggi
5
6  TURNO   DC.B      1          *flag di attivazione dei due nodi
7  FASE    DC.B      0          *flag per la definizione della fase
8  COUNT_CAR DC.B      0        *contatore per i caratteri
9  COUNT_MESS DC.B      0        *contatore per i messaggi
10 MESSAGE DS.B      N          *messaggio da N caratteri
11
12 * ----- MAIN -----
13      ORG      $8200
14  PIA1D   EQU      $2002      *registro dato di PIA1
15  PIA1C   EQU      $2003      *registro controllo di PIA1
16  PIA2D   EQU      $2004      *registro dato di PIA2
17  PIA2C   EQU      $2005      *registro controllo di PIA2
18
19  MAIN    JSR      SETUP_PIA
20          MOVE.W   SR,D0      *salvo SR in D0 per modificarlo
21          ANDI.W   #$D8FF,D0  *imposto lo stato utente in D0
22          MOVE.W   D0,SR      *pongo il nuovo SR nel registro
23  LOOP    END      LOOP      *ciclo caldo di attesa di interruzioni
24
25  SETUP_PIA MOVE.B   #0,PIA1C  *ricezione con PIA1
26          MOVE.B   #$FF,PIA1D
27          MOVE.B   #%00100101,PIA1C
28
29          MOVE.B   #0,PIA2C    *ricezione con PIA2
30          MOVE.B   #$FF,PIA2D
31          MOVE.B   #%00100101,PIA2C
32  RTS
33
```

```

34 * ----- INTERRUZIONE B -----
35      ORG      $8500
36 ISR_B      MOVEA.L  A0,-(A7)      *SETTAGGIO DEL CONTESTO DELLA ISR
37            MOVEA.L  A1,-(A7)      *  pomendo all'interno dello stack
38            MOVEA.L  A2,-(A7)      *  tutti i registri che si vanno
39            MOVEA.L  A3,-(A7)      *  a sporcare
40            MOVEA.L  A4,-(A7)
41            MOVE.L   D0,-(A7)
42            MOVEA.B  #PIA1D,A0      *DEFINIZIONE DEGLI INDIRIZZI
43            MOVEA.B  #PIA1C,A1      *  delle PIA che si useranno
44            MOVEA.B  #PIA2D,A2      *  nella ISR
45            MOVEA.B  #PIA1C,A3
46            MOVEA.B  #MESSAGE,A4    *DEFINIZIONE DELL'INDIRIZZO DEL MESSAGGIO
47            MOVE.B   FASE,D0
48            CMP.B    #0,D0          *CHECK PER VEDERE SE SONO IN FASE 1
49            BNE      ELSE
50            MOVE.B   COUNT_CAR,D0
51            MOVE.B   (A0),(A4,D0)    *LETTURA DEL CARATTERE DALLA PIA1
52            ADD.B    #1,D0          *  aggiorno il numero di caratteri
53            MOVE.B   D0,COUNT_CAR
54            CMP.B    #N,COUNT_CAR
55            BNE      FINE          * se i caratteri non sono N non  terminato
56            MOVE.B   #0,D0          *      il messaggio e quindi continuo
57            MOVE.B   D0,COUNT_CAR
58            MOVE.B   COUNT_MESS,D0  *  aggiorno il numero di messaggi ricevuti se
59            ADD.B    #1,D0          *      i caratteri sono N
60            CMP.B    #2,D0
61            BNE      FINE          * se ho ricevuto 2 messaggi posso riattivare C
62            MOVE.B   FASE,D0        *      e passare alla fase 2
63            MOVE.B   #1,D0
64            MOVE.B   D0,FASE
65            MOVE.B   TURNO,D0
66            MOVE.B   #1,D0          * RIATTIVAZIONE DEL NODO C
67            MOVE.B   D0,TURNO
68            MOVE.B   COUNT_MESS,D0
69            CMP.B    #M,D0
70            BNE      CHECKCRA7_1
71            MOVE.B   #$00,(A1)      *DISATTIVAZIONE PIA1 se ho ricevuto M messaggi
72 CHECKCRA7_1 MOVE.B   (A3),D0        * controllo se c'e' stata una richiesta di
73            ANDI.B   #$80,D0        *  interruzione da parte del nodo C
74            CMP.B    #$80,D0
75            BNE      FINE          *se non ho avuto interruzioni procedo a terminare
76            la ISR
77            MOVE.B   COUNT_CAR,D0
78            MOVE.B   (A2),(A4,D0)    *  altrimenti leggo il carattere da C
79            ADD.B    #1,D0
80            MOVE.B   D0,COUNT_MESS
81 ELSE      BRA      FINE
82            MOVE.B   TURNO,D0        *SE NON SONO IN FASE 1
83            CMP.B    #1,D0          *  se e' il turno di B leggo il carattere dalla
84            sua PIA
85            BNE      FINE
86            MOVE.B   COUNT_CAR,D0
87            MOVE.B   (A0),(A4,D0)    *LETTURA DEL CARATTERE DALLA PIA1

```

```

86      ADD.B      #1,D0
87      MOVE.B     D0,COUNT_CAR
88      CMP.B      #N,D0
89      BNE        FINE      *se non ho terminato il messaggio vado a FINE
90      MOVE.B     #0,D0      * altrimenti aggiorgno il numero di messaggi
                                ricevuti
91      MOVE.B     D0,COUNT_CAR
92      MOVE.B     COUNT_MESS,D0
93      ADD.B      #1,D0
94      MOVE.B     D0,COUNT_MESS
95      MOVE.B     #0,D0
96      MOVE.B     D0,TURN0   *ATTIVAZIONE DEL NODO C TRAMITE IL FLAG TURNO
97      MOVE.B     COUNT_MESS,D0
98      CMP.B      #M,D0
99      BNE        CHECKCRA7_2
100     CHECKCRA7_2 MOVE.B     #$00,(A1) *DISATTIVAZIONE PIA1 se ho ricevuto M messaggi
101     CHECKCRA7_2 MOVE.B     (A3),D0   *controllo se C non ha richiesto interruzioni
102     ANDI.B     #$80,D0      * in tal caso devo servirla e leggere dalla
                                PIA2
103     CMP.B      #$80,D0
104     BNE        FINE
105     MOVE.B     COUNT_CAR,D0
106     MOVE.B     (A2),(A4,D0) *LETTURA DEL CARATTERE DALLA PIA2
107     ADD.B      #1,D0
108     MOVE.B     D0,COUNT_MESS
109     FINE      MOVE.L     (A7)+,D0   *RIPRISTINO DEL CONTESTO AZZERANDO LO STACK
110     MOVE.L     (A7)+,A4      * riporto i registri allo stato originale
111     MOVEA.L    (A7)+,A3      * prima della ISR, deallocandoli dallo stack
112     MOVEA.L    (A7)+,A2
113     MOVEA.L    (A7)+,A1
114     MOVEA.L    (A7)+,A0
115     RTE
116
117     * ----- INTERRUZIONE C -----
118     ORG        $8700
119     ISR_C      MOVEA.L    A0,-(A7)
120     MOVEA.L    A1,-(A7)
121     MOVEA.L    A2,-(A7)
122     MOVEA.L    A3,-(A7)
123     MOVEA.L    A4,-(A7)
124     MOVE.L     D0,-(A7)
125     MOVEA.B    #PIA1D,A0
126     MOVEA.B    #PIA1C,A1
127     MOVEA.B    #PIA2D,A2
128     MOVEA.B    #PIA1C,A3
129     MOVEA.B    #MESSAGE,A4
130     MOVE.B     TURN0,D0
131     CMP.B      #0,D0
132     BNE        FINE
133     MOVE.B     COUNT_CAR,D0
134     MOVE.B     (A2),(A4,D0)
135     ADD.B      #1,D0
136     MOVE.B     D0,COUNT_CAR
137     CMP.B      #N,D0

```

138		BNE	FINE
139		MOVE.B	#0,D0
140		MOVE.B	D0,COUNT_CAR
141		MOVE.B	COUNT_MESS,D0
142		ADD.B	#1,D0
143		MOVE.B	D0,COUNT_MESS
144		MOVE.B	#0,D0
145		MOVE.B	D0,TURN0
146		MOVE.B	COUNT_MESS,D0
147		CMP.B	#M,D0
148		BNE	CHECKCRA7
149		MOVE.B	#\$00,(A3)
150	CHECKCRA7	MOVE.B	(A1),D0
151		ANDI.B	#\$80,D0
152		CMP.B	#\$80,D0
153		BNE	FINE
154		MOVE.B	COUNT_CAR,D0
155		MOVE.B	(A0),(A4,D0)
156		ADD.B	#1,D0
157		MOVE.B	D0,COUNT_MESS
158	FINE	MOVE.L	(A7)+,D0
159		MOVE.L	(A7)+,A4
160		MOVEA.L	(A7)+,A3
161		MOVEA.L	(A7)+,A2
162		MOVEA.L	(A7)+,A1
163		MOVEA.L	(A7)+,A0
164		RTE	
165			
166		END	MAIN

Cosa cambia con il DMA?

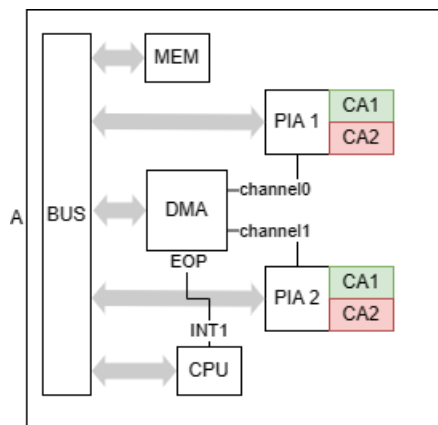
Il **DMA** è un dispositivo in grado di regolare la comunicazione tra le periferiche e l'esterno. Esso si occupa di gestire il bus interno e fare sì che si possano risolvere i conflitti tra CPU e periferiche esterne, quando queste devono porre qualcosa in memoria, o anche quando la CPU deve regolare il trasferimento memoria-memoria. In particolare, esso può svolgere due modalità di funzionamento:

- **SINGLE**: modalità che fa sì che il DMA rilasci il bus alla fine della trasmissione di un singolo carattere;
- **BLOCK**: modalità che fa sì che il DMA occupi il bus per tutta la trasmissione di un intero messaggio.

Nel caso in esame, i nodi A,B e C beneficiano di tale dispositivo, visto che sarà il DMA stesso a segnalare che è terminato un messaggio, e quindi è possibile ragionare non più a caratteri ma a messaggi.

Nello specifico, il DMA internamente fa le veci della CPU per quel che riguarda le interruzioni verso le periferiche esterne.

Aggiungendo il DMA l'architettura del nodo A cambia così come mostrato in figura.



A livello implementativo quello che cambia con l'introduzione del DMA è che all'interno del programma principale è necessario andare ad effettuare la configurazione anche del DMA, visto che la CPU deve comunque inizializzare sia le periferiche che il DMA. Tale configurazione comprende:

- settaggio del CCOUNT per il numero di byte totale;
- definizione del CADDR per l'indirizzo di base;
- definizione del MODE per impostare la modalità di trasferimento periferica-memoria in modalità BLOCK;
- abilitazione del DMA tramite il CONTROL register;
- scrittura dello STATO nello STATUS register.