Nome: Benfenati Domenico

Prova CSD
20 Dicembre 2022 (A-)

Traccia

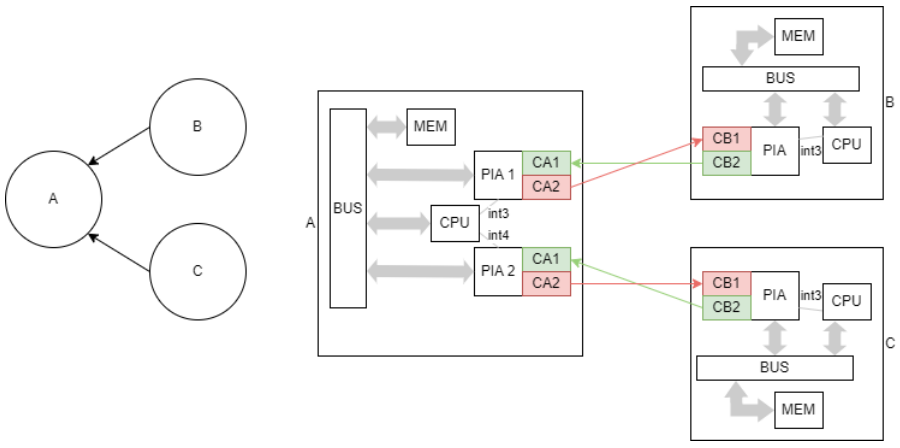
Un sistema è composto da 3 unità, A, B e C. A è collegato in ricezione a B e C mediante due periferiche parallele P1 e P2. B e C sono programmati per inviare continuamente messaggi da N caratteri ad A, ma la ricezione deve essere gestita in modo tale da assicurare che **in ogni istante il numero di messaggi 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, proponento 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.

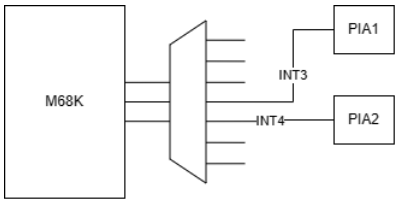
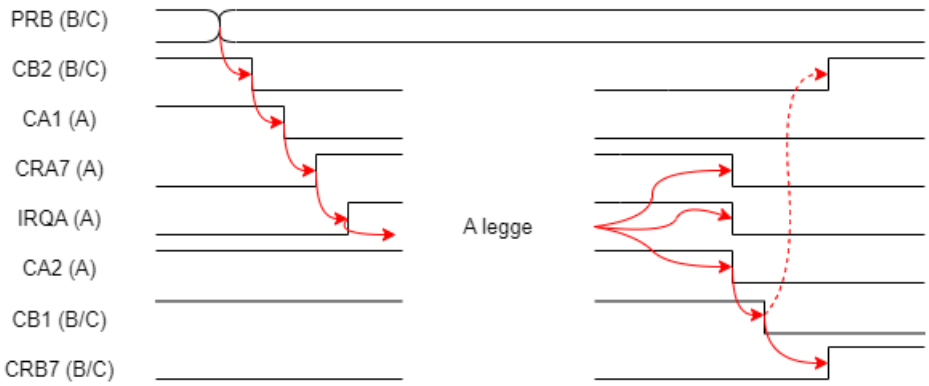


Figura 1: Architettura delle interruzioni nel nodo A

Protocolli

La comunicazione tra A e i nodi trasmittenti 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	
$8000	TURNO	FASE
$8002	COUNT_CAR	COUNT_MESS
$8004	MESSAGGIO	MESSAGGIO
$8006	MESSAGGIO	MESSAGGIO
	...	
$8200	MAIN	
	...	
$8500	ISR_B	
	...	
$8700	ISR_C	
	...	

	MEMORIA ROM
$6C	00 00 85 00
$70	00 00 87 00

Pseudocodice

Per fare si 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.

```
MAIN(){
    inizializza PIA 1 in ricezione;
    inizializza PIA 2 in ricezione;
    Enable stato utente;
    Enable interruzioni;
    while(true);            //attendo l'interruzione
}

/*
FLAG UTILIZZATI:
- 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
- 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.
*/
ISR_B(){
    if(FASE==0){
        leggi da PIA1;
        count_car ++;
        if(count_car == N){
            count_car = 0;
            count_mess ++;
            if(count_mess == 2){
                FASE = 1;
                TURNO = 1;
                if(CRA7(C) == 1){
                    leggi da PIA2;
                    count_car ++;
                }
            }
        }
    }
    else if(TURNO == 1){    //Se non e' il turno di B, B non puo' mandare
        leggi da PIA1;
        count_car ++;
        if(count_car == N){ //termina l'invio di un messaggio da B
            count_car = 0;
            count_mess ++;
            TURNO = 0;      //riattivazione di C
```

```
            if(count_mess == M){
                disattiva PIA1;
            }
            if(CRA7(B) == 1){
                leggi da PIA2;
                count_car ++;
            }
        }
    }
    return from ISR;
}

ISR_C(){
    if(TURNO == 0){
        leggi da PIA2;
        count_car ++;
        if(count_car == N){
            count_car = 0;
            count_mess ++;
            TURNO = 1;      //riattivazione di B
            if(count_mess == M){
                disattivazione PIA2;
            }
            if(CRA7(B) == 1){
                leggi da PIA1;
                count_car ++;
            }
        }
    }
    return from ISR;
}
```

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 MESSAGGE    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
```

```
* --------------------- INTERRUZIONE B ----------------------------
            ORG         $8500
ISR_B       MOVEA.L     A0,-(A7)        *SETTAGGIO DEL CONTESTO DELLA ISR
            MOVEA.L     A1,-(A7)        *   pomendo all'interno dello stack
            MOVEA.L     A2,-(A7)        *   tutti i registri che si vanno
            MOVEA.L     A3,-(A7)        *   a sporcare
            MOVEA.L     A4,-(A7)
            MOVE.L      D0,-(A7)
            MOVEA.B     #PIA1D,A0       *DEFINIZIONE DEGLI INDIRIZZI
            MOVEA.B     #PIA1C,A1       *   delle PIA che si useranno
            MOVEA.B     #PIA2D,A2       *   nella ISR
            MOVEA.B     #PIA1C,A3
            MOVEA.B     #MESSAGE,A4     *DEFINIZIONE DELL'INDIRIZZO DEL MESSAGGIO
            MOVE.B      FASE,D0
            CMP.B       #0,D0           *CHECK PER VEDERE SE SONO IN FASE 1
            BNE         ELSE
            MOVE.B      COUNT_CAR,D0
            MOVE.B      (A0),(A4,D0)    *LETTURA DEL CARATTERE DALLA PIA1
            ADD.B       #1,D0           * aggiorno il numero di caratteri
            MOVE.B      D0,COUNT_CAR
            CMP.B       #N,COUNT_CAR
            BNE         FINE            * se i caratteri non sono N non    terminato
            MOVE.B      #0,D0           *       il messaggio e quindi continuo
            MOVE.B      D0,COUNT_CAR
            MOVE.B      COUNT_MESS,D0   * aggiorno il numero di messaggi ricevuti se
            ADD.B       #1,D0           *       i caratteri sono N
            CMP.B       #2,D0
            BNE         FINE            * se ho ricevuto 2 messaggi posso riattivare C
            MOVE.B      FASE,D0         *       e passare alla fase 2
            MOVE.B      #1,D0
            MOVE.B      D0,FASE
            MOVE.B      TURNO,D0
            MOVE.B      #1,D0           * RIATTIVAZIONE DEL NODO C
            MOVE.B      D0,TURNO
            MOVE.B      COUNT_MESS,D0
            CMP.B       #M,D0
            BNE         CHECKCRA7_1
            MOVE.B      #$00,(A1)       *DISATTIVAZIONE PIA1 se ho ricevuto M messaggi
CHECKCRA7_1 MOVE.B      (A3),D0         * controllo se c'e' stata una richiesta di
            ANDI.B      #$80,D0         *   interruzione da parte del nodo C
            CMP.B       #$80,D0
            BNE         FINE            *se non ho avuto interruzioni procedo a terminare
                 la ISR
            MOVE.B      COUNT_CAR,D0
            MOVE.B      (A2),(A4,D0)    *   altrimenti leggo il carattere da C
            ADD.B       #1,D0
            MOVE.B      D0,COUNT_MESS
            BRA         FINE
ELSE        MOVE.B      TURNO,D0        *SE NON SONO IN FASE 1
            CMP.B       #1,D0           *   se e' il turno di B leggo il carattere dalla
                sua PIA
            BNE         FINE
            MOVE.B      COUNT_CAR,D0
            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,TURNO        *ATTIVAZIONE DEL NODO C TRAMITE IL FLAG TURNO
97              MOVE.B      COUNT_MESS,D0
98              CMP.B       #M,D0
99              BNE         CHECKCRA7_2
100             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      TURNO,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
```

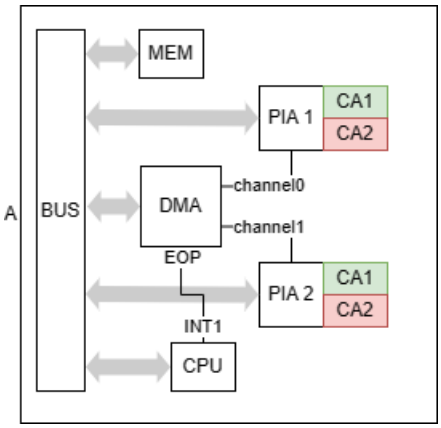
```
138             BNE         FINE
139             MOVE.B      #0,D0
140             MOVE.B      D0,COUNT_CAR
141             MOVE.B      COUNT_MESS,D0
142             ADD.B       #1,D0
143             MOVE.B      D0,COUNT_MESS
144             MOVE.B      #0,D0
145             MOVE.B      D0,TURNO
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 si che si possano risolvere i conflitti tra CPU e preiferiche 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.

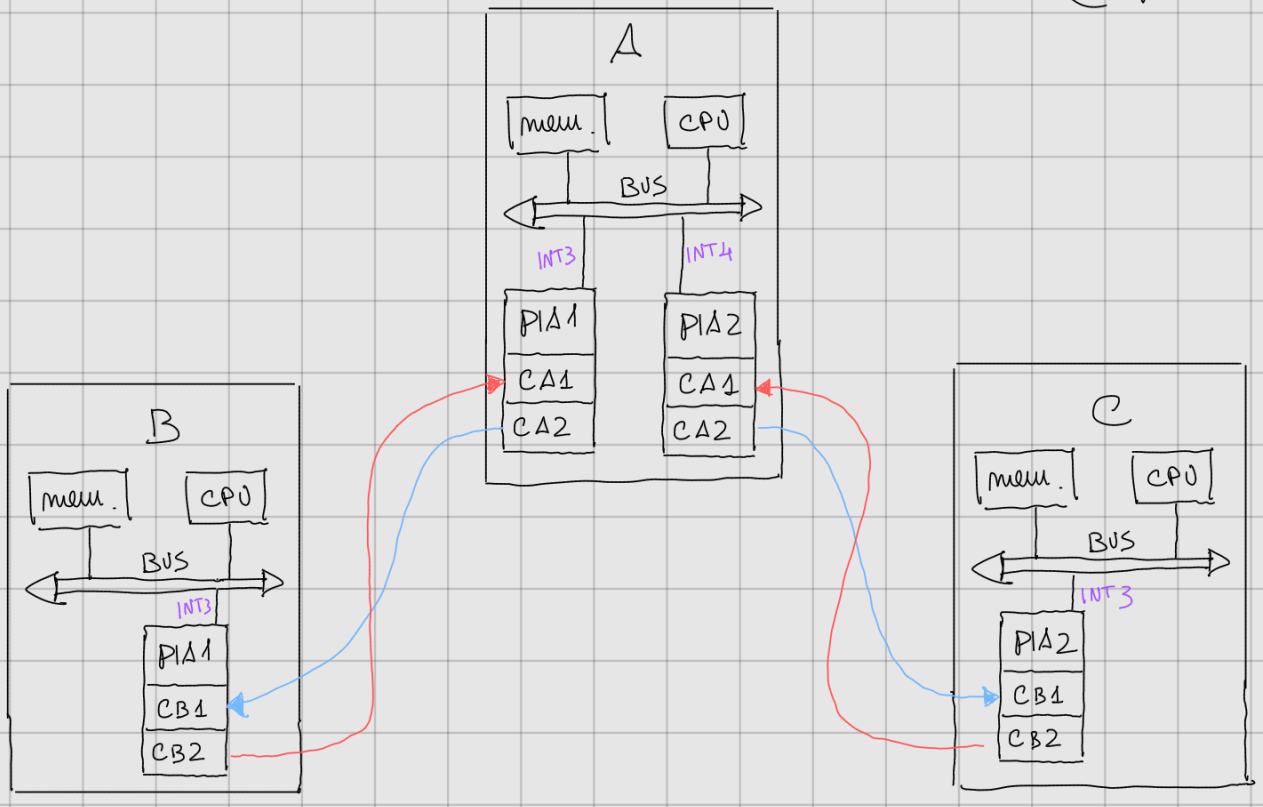


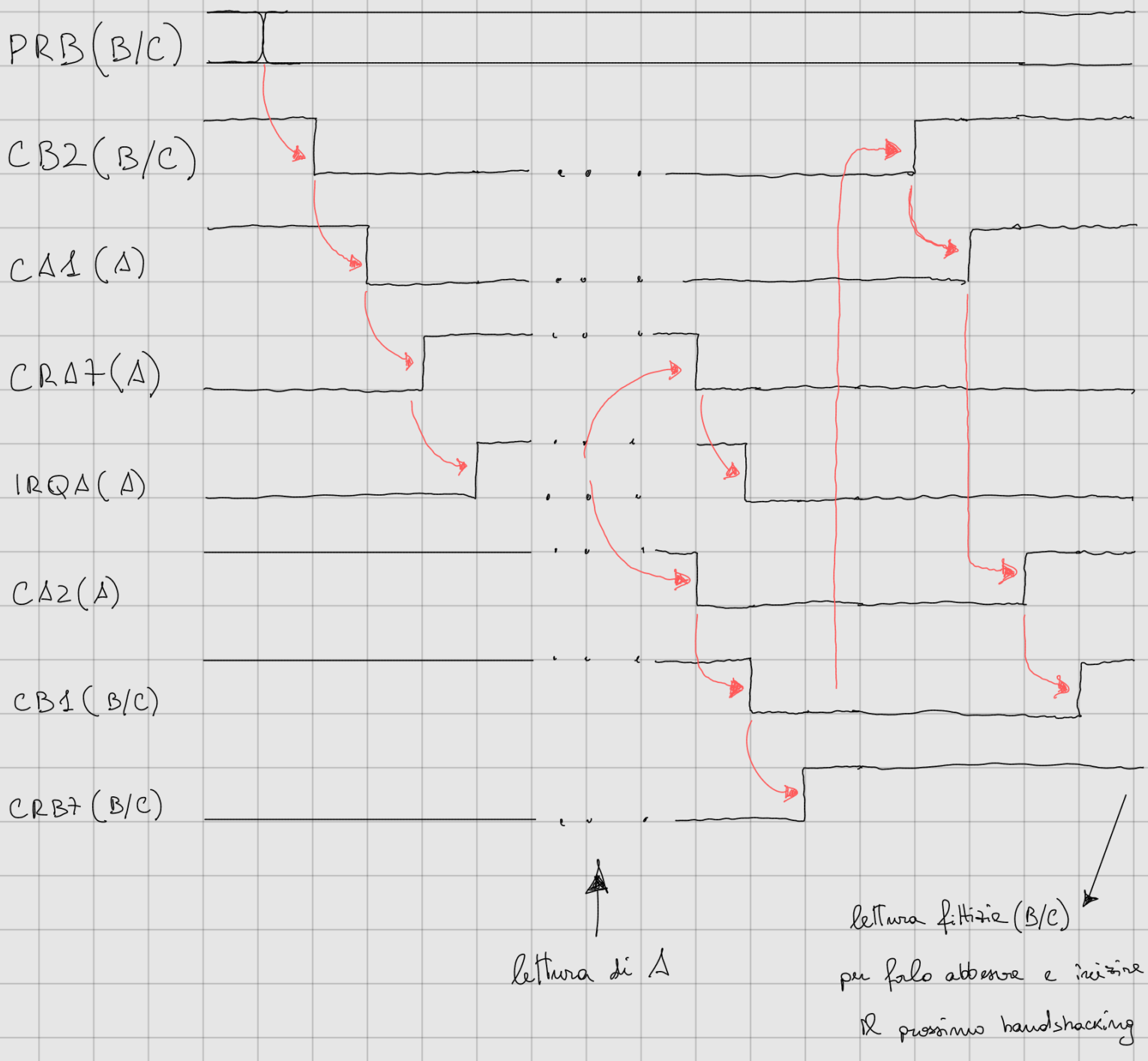
A livello implementativo quello che cambia con l'introduzione del DMA è che all'interno del programma principale è necessario andare ad effettuare la configurazione anche del DMA, visto che la CPU deve comunque inizializzare sia le periferiche che il DMA. Tale configurazione comprende:

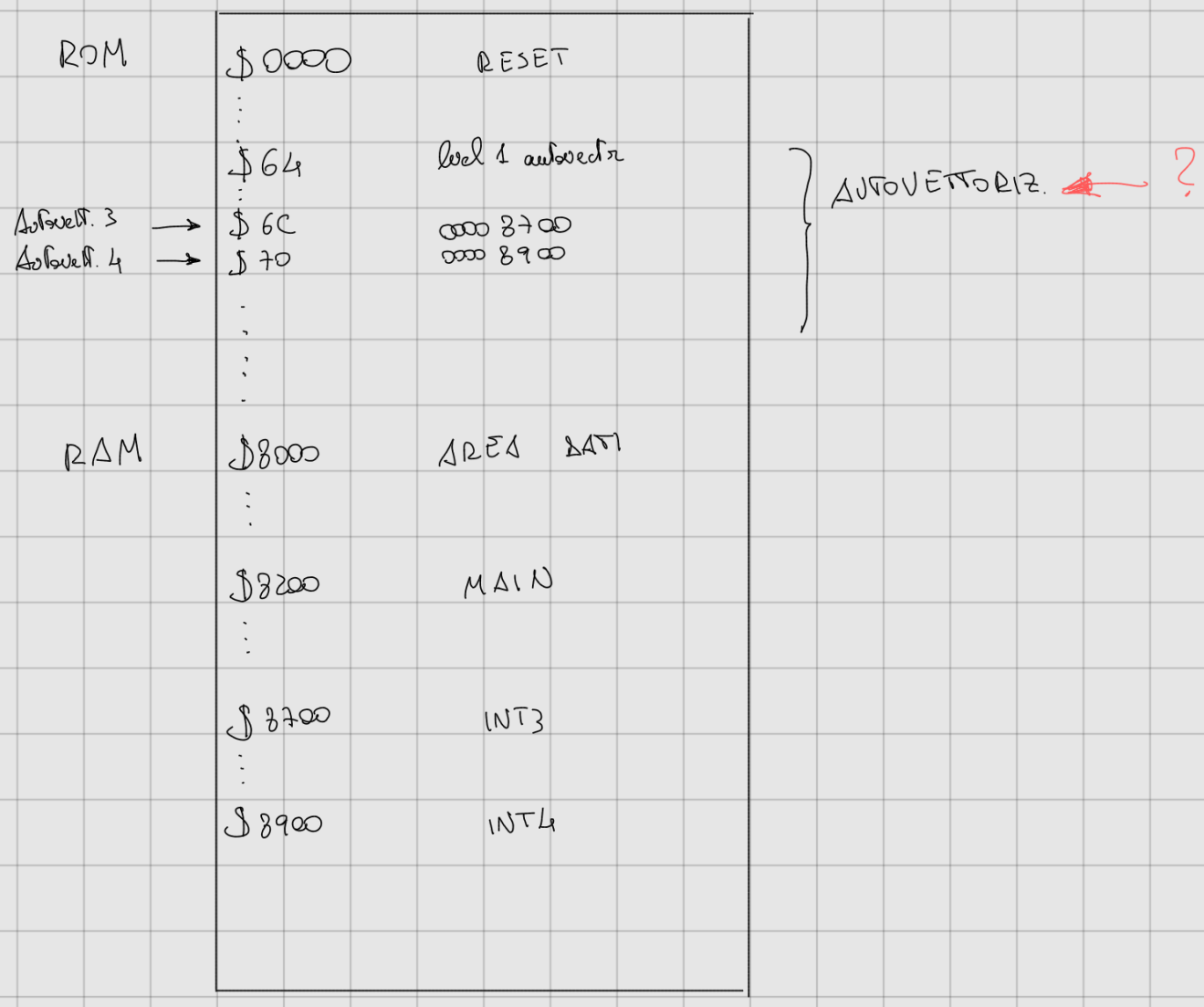
- settaggio del CCOUNT per il numero di byte totale;
- definizione del CADDR per l'indirizzo di base;
- definizione del MODE per impostare la modalità di trasferimento periferica-memoria in modalità BLOCK;
- abilitazione del DMA tramite il CONTROL register;
- scrittura dello STATO nello STATUS register.

- B, C inviano N caratteri (1 messaggio) ad A
- In ogni istanti mess. ricevuti da B > mess. ricevuti da C

B ↑ priorità
C ↓ "




PRB (B/C)
CB2 (B/C)
CA1 (A)
CRA7 (A)
IRQA (A)
CA2 (A)
CB1 (B/C)
CRB7 (B/C)
lettura di A
lettura fittizia (B/C)
per fare abbassare e iniziare
il prossimo handshacking


ROM
$0000
RESET
$64
livel 1 autovettr
Autovett. 3
$6C
0000 8700
Autovett. 4
$70
0000 8900
AUTOVETTORIZ.
?
RAM
$8000
AREA DATI
$8200
MAIN
$8700
INT3
$8900
INT4

NODO A