Nome: Benfenati Domenico

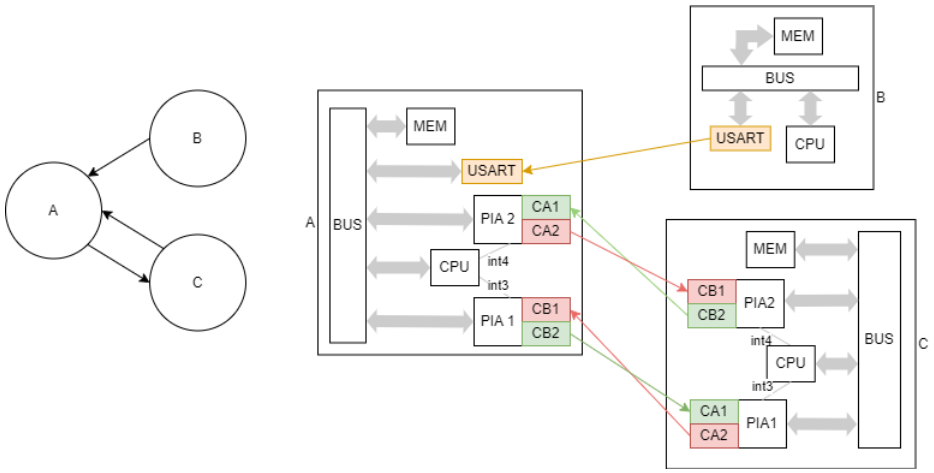
**Prova CSD**
**11 Gennaio 2022**

## Traccia

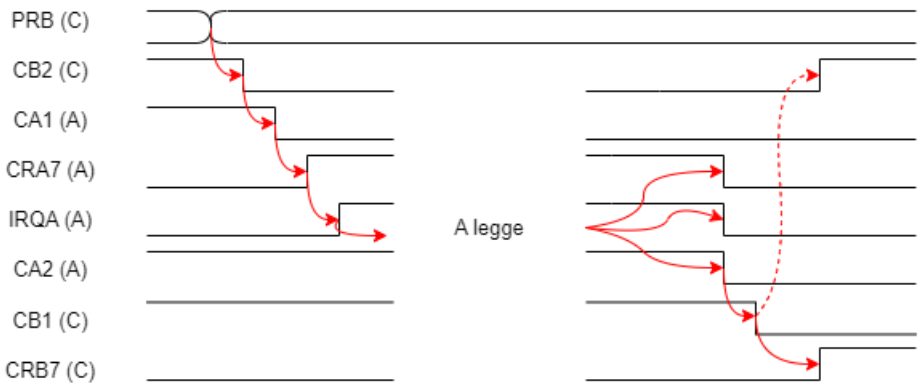
Un sistema è composto da 3 unità: A, B e C. A è collegata a B mediante una periferica seriale e a C mediante 2 periferiche parallele, una P1 configurata in trasmissione e l'altra P2 in ricezione. Il sistema opera come segue:
A riceve da B una sequenza di M messaggi da N caratteri ciascuno attraverso la periferica seriale. Dopo aver ricevuto interamente un messaggio, A lo inoltra a C mediante la parallela P1 e riceve successivamente da C un carattere di ACK ($FF) al termine del trasferimento, mediante P2.
Si progetti e implementi l'unità A ponendosi in una delle seguenti ipotesi:

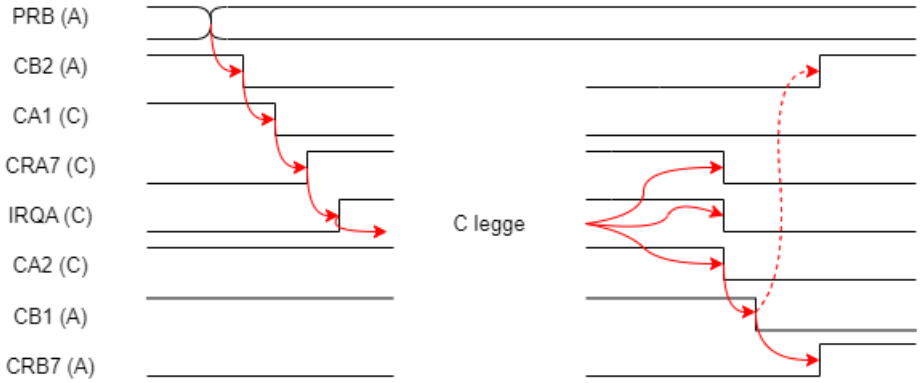
- A non inizia a ricevere un nuovo messaggio da B se non ha prima interamente inviato il precedente e ricevuto l'ACK da C. In questo caso si può ipotizzare che anche fra A e B sia presente una periferica parallela al posto della seriale.
- A non inizia a ricevere un nuovo messaggio da B se non ha prima interamente inviato il precedente e ricevuto l'ACK da C. In questo caso è necessario prevedere un controllo di fl usso esplicito fra A e B per la ricezione mediante seriale.
- A può ricevere nuovi messaggi da B anche prima di aver terminato la trasmissione dei precedenti e di aver ricevuto il relativo ACK. In questo caso è necessario individuare e gestire opportunamente situazioni di conflitto.

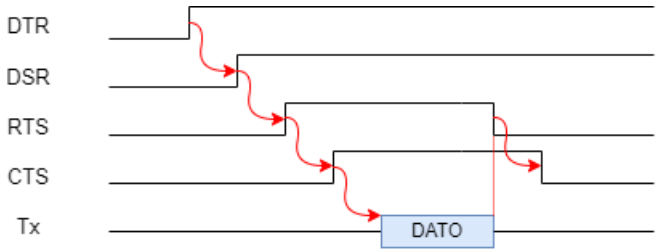
## Architettura



# Protocolli


PRB (C)
CB2 (C)
CA1 (A)
CRA7 (A)
IRQA (A)
CA2 (A)
CB1 (C)
CRB7 (C)
A legge


PRB (A)
CB2 (A)
CA1 (C)
CRA7 (C)
IRQA (C)
CA2 (C)
CB1 (A)
CRB7 (A)
C legge


DTR
DSR
RTS
CTS
Tx
DATO

## Mappa della memoria

| | MEMORIA RAM | |
|---|---|---|
| $8000 | COUNT_CAR | COUNT_MESS |
| $8002 | CHECK | MESSAGE |
| $8004 | MESSAGE | MESSAGE |
| $8006 | MESSAGE | |
| | ... | |
| $8100 | MAIN | |
| | ... | |
| $8500 | ISR_B | |
| | ... | |

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

## Pseudocodice

Per tale esercizio si è scelto di gestire l'**ipotesi 3**, e quindi regolare lo scambio di messaggi gestendo gli eventuali conflitti tra le ISR dei nodi.
Si è scelto di risolvere tali situazioni mediante l'uso di appositi flag per la gestione delle fasi operative del nodo A. L'uso di tali flag in maniera opportuna rende inutile l'introduzione di un eventuale TAS.

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

/*
FLAG UTILIZZATI:
- ricevi: e' TRUE se sono nella fase di ricezione dall'USART di B
- trasmetti: e' TRUE se sono nella fase di invio del messaggio a C tramite PIA1
- waitAck: e' TRUE se sono nella fase di attesa del messaggio di ACK da C tramite PIA2
*/
ISR_B(){
    if(ricevi==true AND trasmetti==false AND waitAck==false){
        Ricevi da USART;
        count_car ++ ;
        if(count_car == N){
            count_car = 0;
            count_mess ++ ;
            trasmetti = true;
            ricevi = false;
            waitAck = false;
            if(count_mess == M){
                disattiva USART;
            }
        }
```

```
    }
    return from ISR;
}

ISR_C(){
    if(ricevi==false AND trasmetti==true AND waitAck==false){
        Invia su PIA1;
        count_send ++ ;
        if(count_send == N){
            trasmetti = false;
            ricevi = false;
            waitAck = true;
        }
    }
    return from ISR;
}

ISR_ACK(){
    if(ricevi==false AND trasmetti==false AND waitAck==true){
        Ricevi da PIA2;
        trasmetti = false;
        ricevi = true;
        waitAck = false;
        if(CRA7(B) == 1){
            Leggi da USART;
            count_car ++ ;
        }
    }
    return from ISR;
}
```

## Programma assembly

```
            ORG     $8000
N           EQU     4
M           EQU     10

COUNT_CAR   DC.B    0
COUNT_MESS  DC.B    0
TRASMETTI   DC.B    0
RICEVI      DC.B    1
WAITACK     DC.B    0
MESSAGGE    DS.B    N

            ORG     $8200
USARTD      EQU     $2000   *registro dato dell'USART
USARTC      EQU     $2001   *registro modo-controllo dell'USART
PIA1D       EQU     $2004   *registro dato di PIA1
PIA1C       EQU     $2005   *registro controllo di PIA1
PIA2D       EQU     $2006   *registro dato di PIA2
PIA2C       EQU     $2007   *registro controllo di PIA2

```

```
MAIN        JSR     SETUP_PIA
            JSR     SET_USART
            MOVE.W  SR,D0       *salvo SR in D0 per modificarlo
            ANDI.W  #$D8FF,D0   *imposto lo stato utente e le interruzioni in D0
            MOVE.W  D0,SR       *pongo il nuovo SR nel registro
LOOP        END     LOOP

SETUP_PIA   MOVE.B  #0,PIA1C            *invio con PIA1
            MOVE.B  #$00,PIA1D
            MOVE.B  #%00100101,PIA1C

            MOVE.B  #0,PIA2C            *ricevo con PIA2
            MOVE.B  #$FF,PIA2D
            MOVE.B  #%00100101,PIA2C
            RTS

SET_USART   MOVE.B  #%01011101,USARTC   *setto msg da 8 bit, comunicazione async
            MOVE.B  #%00110110,USARTC   *imposto l'USART in ricezione
            RTS

            ORG     $8500
ISR_B       MOVEA.L A0,-(A7)
            MOVEA.L A1,-(A7)
            MOVEA.L A2,-(A7)
            MOVEA.L A3,-(A7)
            MOVE.L  D0,-(A7)
            MOVEA.L A4,-(A7)


FINE        MOVEA.L (A7)+,A4
            MOVE.L  (A7)+,D0
            MOVEA.L (A7)+,A3
            MOVEA.L (A7)+,A2
            MOVEA.L (A7)+,A1
            MOVEA.L (A7)+,A0
            RTE


            END     MAIN
```