PD32

Interfacciamento con i dispositivi di I/O

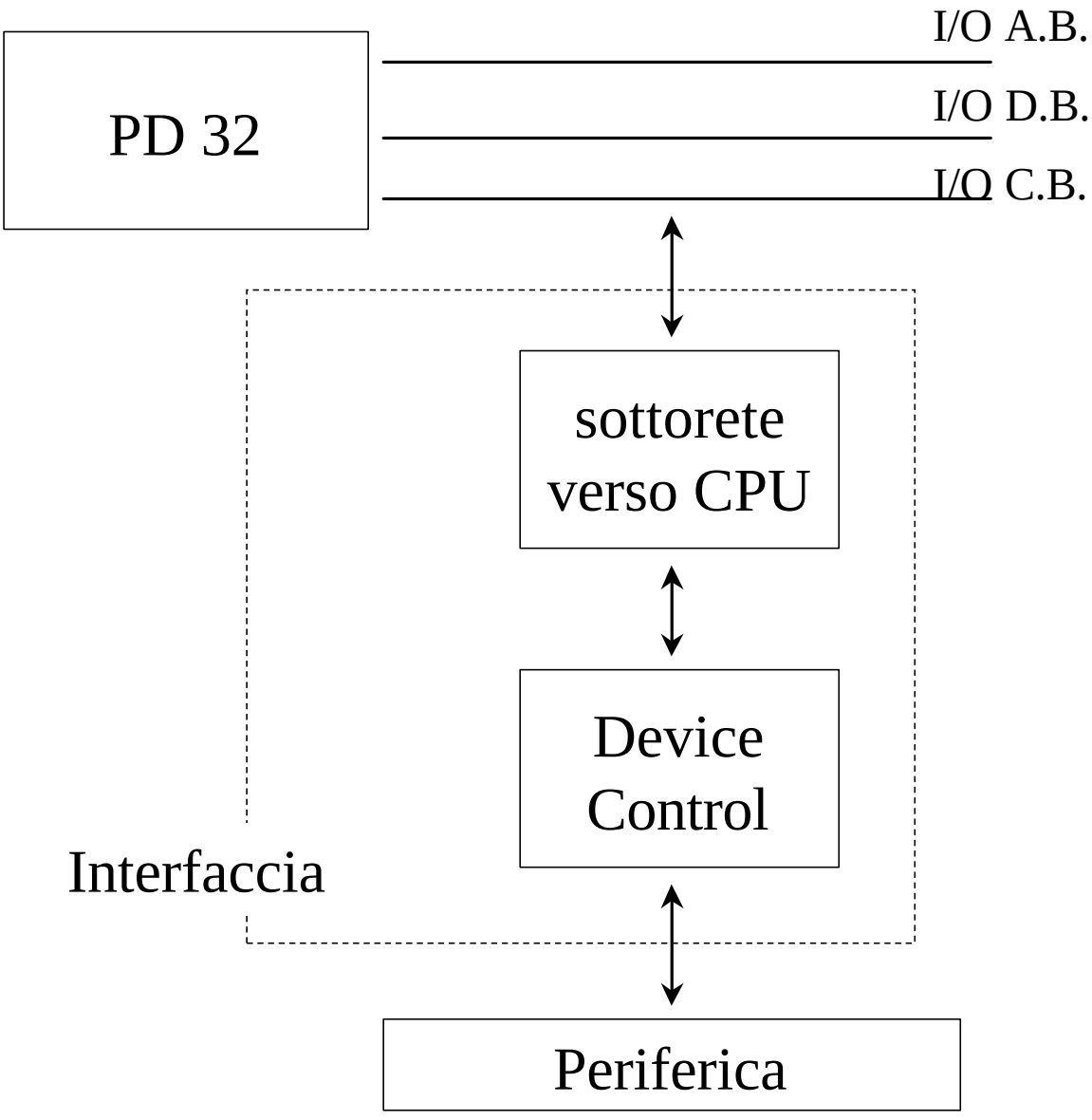
(V)

Interazione CPU - dispositivi

Soluzioni possibili

- Busy Waiting
- Utilizzabile quando CPU esegue solo il task di dialogo con la periferica
- Interruzioni
- D.M.A.

Interfacciamento del PD32



Istruzioni di I/O

Riferendoci ai campi I/O, K del formato istruzione del PD32:

IO=01 -> K = indirizzo del dispositivo

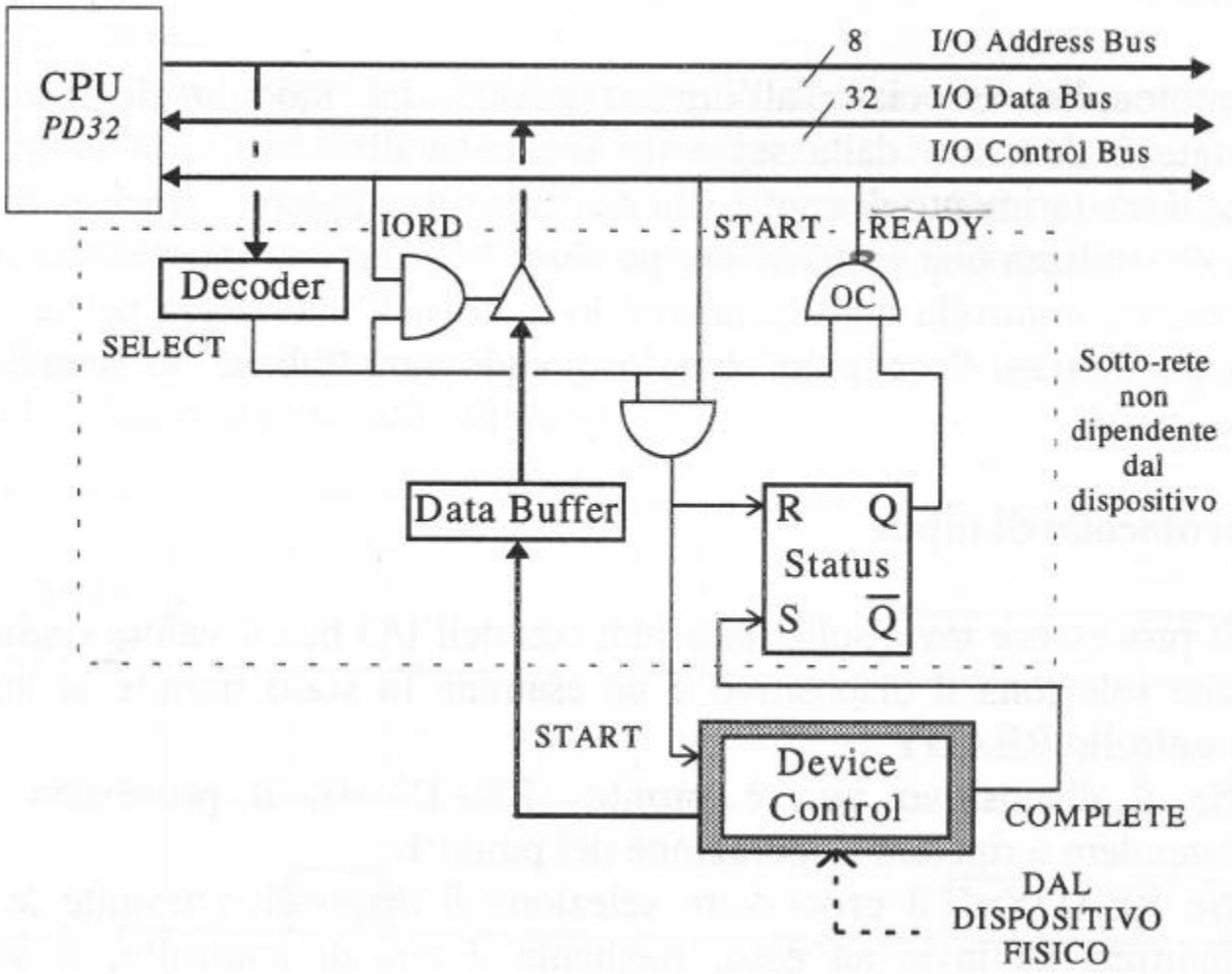
IO=10 -> K {bit 18…16} = registro

TIPO	CODICE	OPERANDI	C N Z V P I	COMMENTO
0	INs	dev , D0	- - - - - -	Il dato contenuto nel buffer del device dev è trasferito nella destinazione D0 dev→D0
1	OUTs	S , dev	- - - - - -	Il dato sorgente S viene trasferito nel buffer del device dev sorg→dev
2	START	dev	- - - - - -	Viene azzerato il flip-flop STATUS del dev e viene avviata l'operazione
3	CLEAR	dev	- - - - - -	Viene azzerato il flip-flop STATUS del dev senza avviare l'operazione
4	JR	dev , D1	- - - - - -	Se STATUS=1 salta alla destinazione D1
5	JNR	dev , D1	- - - - - -	Se STATUS=0 salta alla destinazione D1
6	SETIM	dev	- - - - - -	Viene abilitato il device dev ad inviare interruzioni 1→IM
7	CLRIM	dev	- - - - - -	Viene disabilitato il device dev ad inviare interruzioni 0→IM
8	JIM	dev , D1	- - - - - -	Se IM=1 salta alla destinazione D1
9	JNIM	dev , D1	- - - - - -	Se IM=0 salta alla destinazione D1

Con s si indica il formato del dato: B, W, L.

L’indirizzamento del dispositivo (indirizzo *dev*) può essere soltanto *diretto con registro* o *assoluto*.

Esempio di interfaccia con dispositivo di input



Protocollo di input
Temporizzazioni

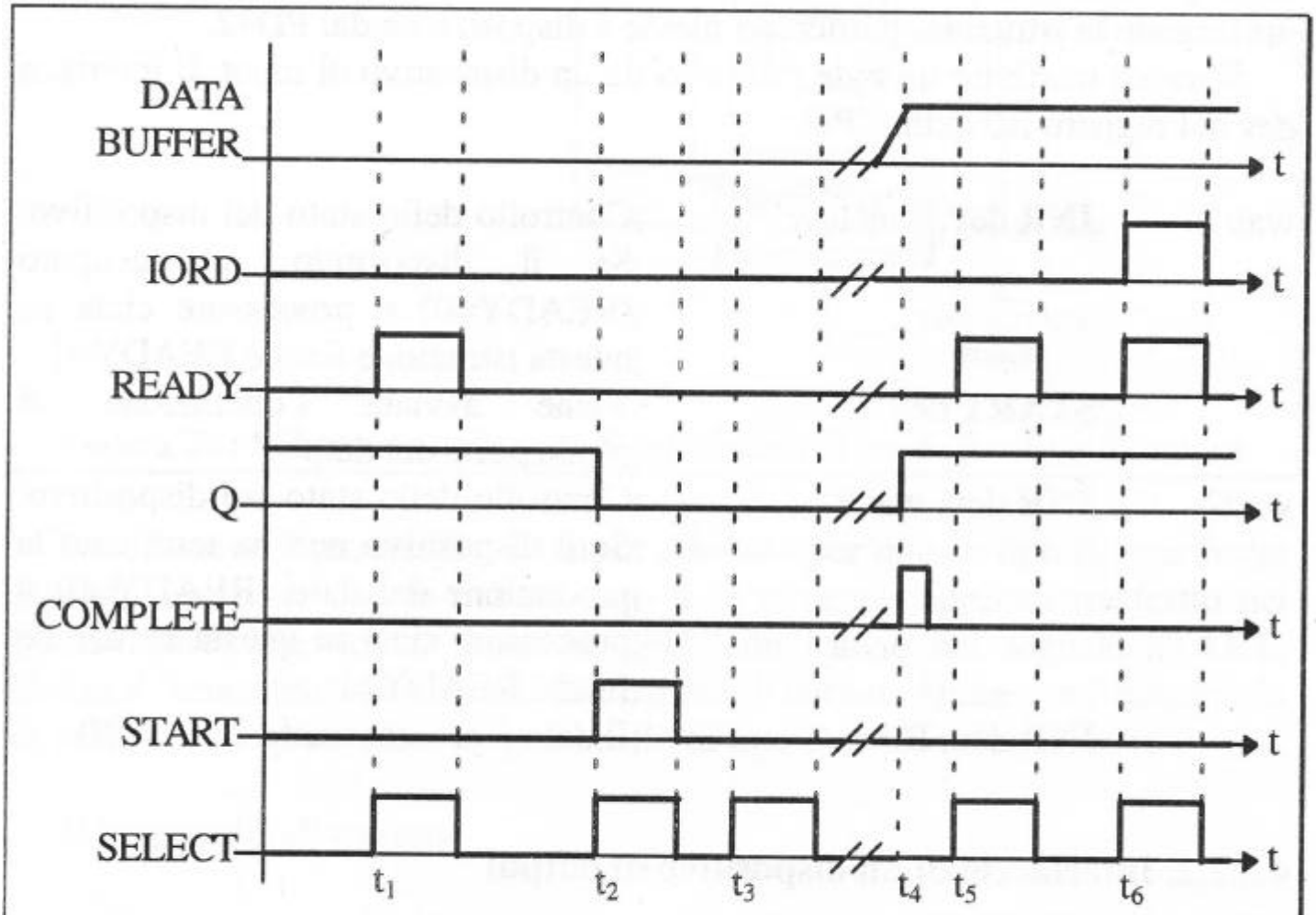
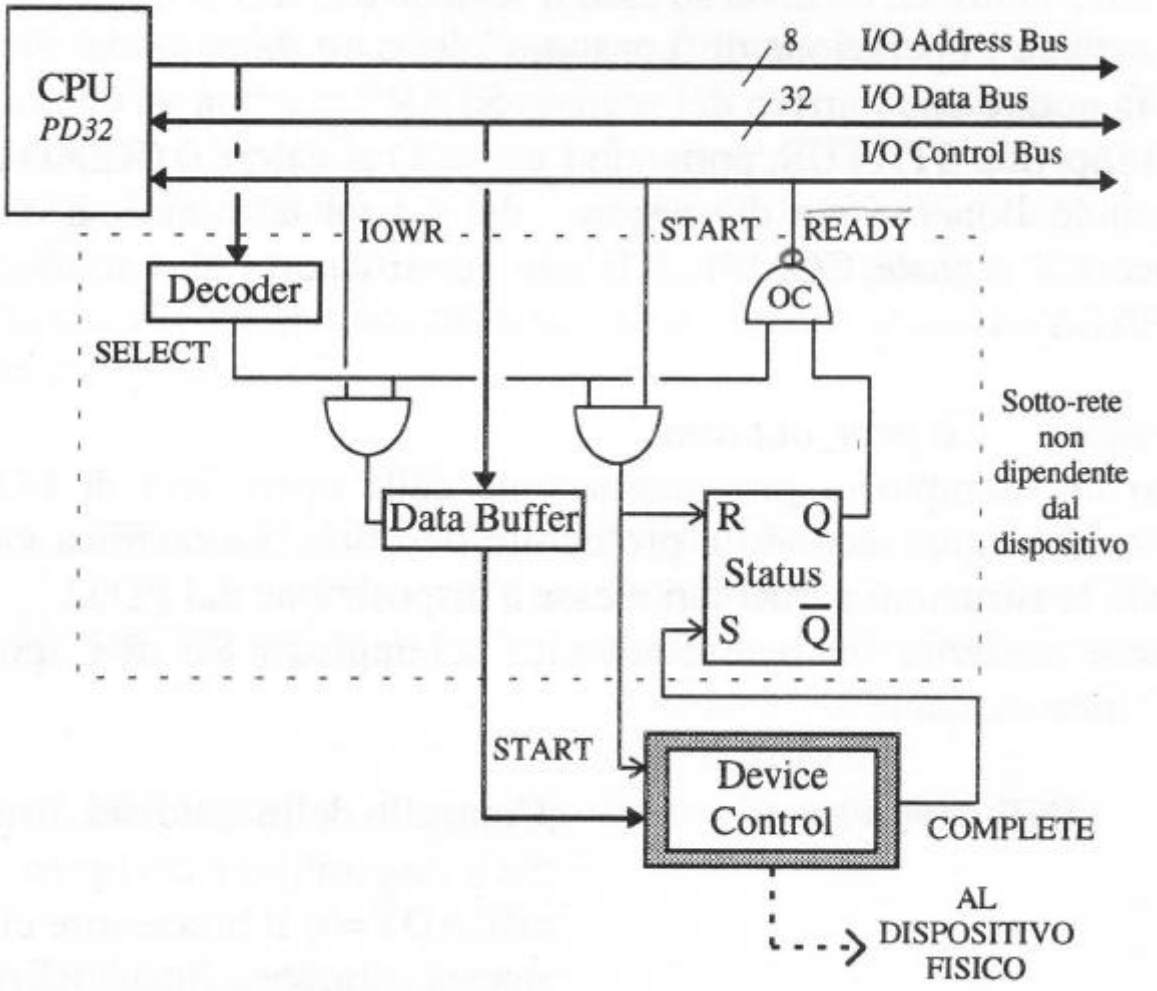


Figura 25: Andamento temporale di alcuni segnali durante il protocollo di input

Esempio di interfaccia con dispositivo di output



Esempio di protocollo di input

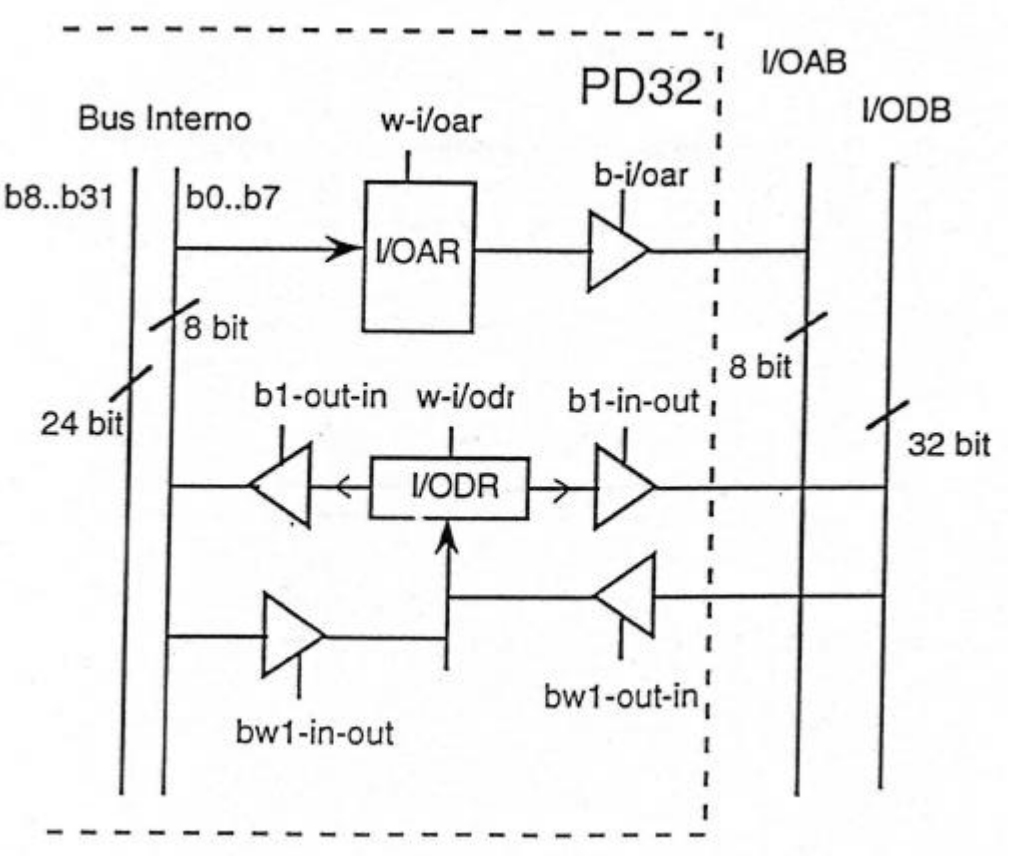
“Si vuole trasferire 5 dati in formato word da un dispositivo di input (indirizzo=dev) in memoria a partire da 800h”

```
                …
                movl #5, r0
                movl #800h, r1
                jsr input
                …

                ; subroutine di input
input:          push r0
                push r1
pronta:         jnr dev, pronta
inizio:         start dev
produce:        jnr dev, produce
                inw dev, (r1)+
                subb #1, r0
                jnz inizio
                pop r1
                pop r0
                ret
```

Interfaccia interna tra PD32 e bus periferiche

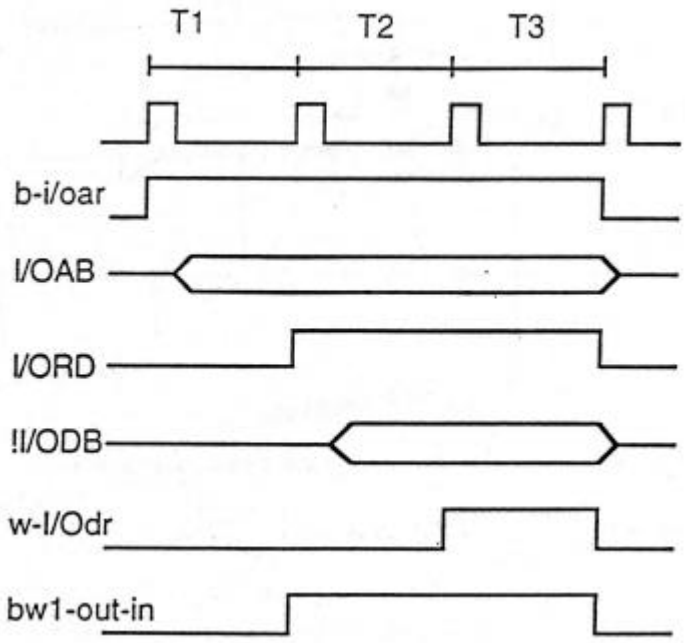
- L’interfaccia è più semplice rispetto al caso della memoria, è basata su registro I/ODR e I/OAR:
- Bus interno -> I/OAR: avviene attraverso abilitazione segnale w-i/oar
- I/OAB-> I/O A.B.: attraverso segnale b-i/oar
- Lettura I/ODR: su bus attraverso b1-out-in, su I/O D.B. attraverso b1-out-in
- Scrittura I/ODR: abilitando w-i/odr e (da bus) bw1-in-out oppure (da I/O D.B.) bw1-out-in



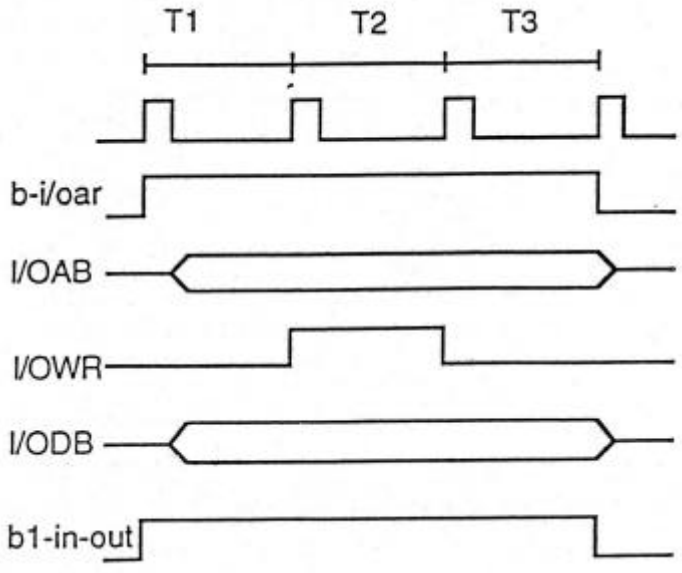
Cicli di Accesso a periferiche

- Ipotesi: tempo produzione/consumo di un dato < 1 ciclo CLK
- Interazione completata in 3 cicli, comandi generati sul fronte di salita del CLK.

INPUT



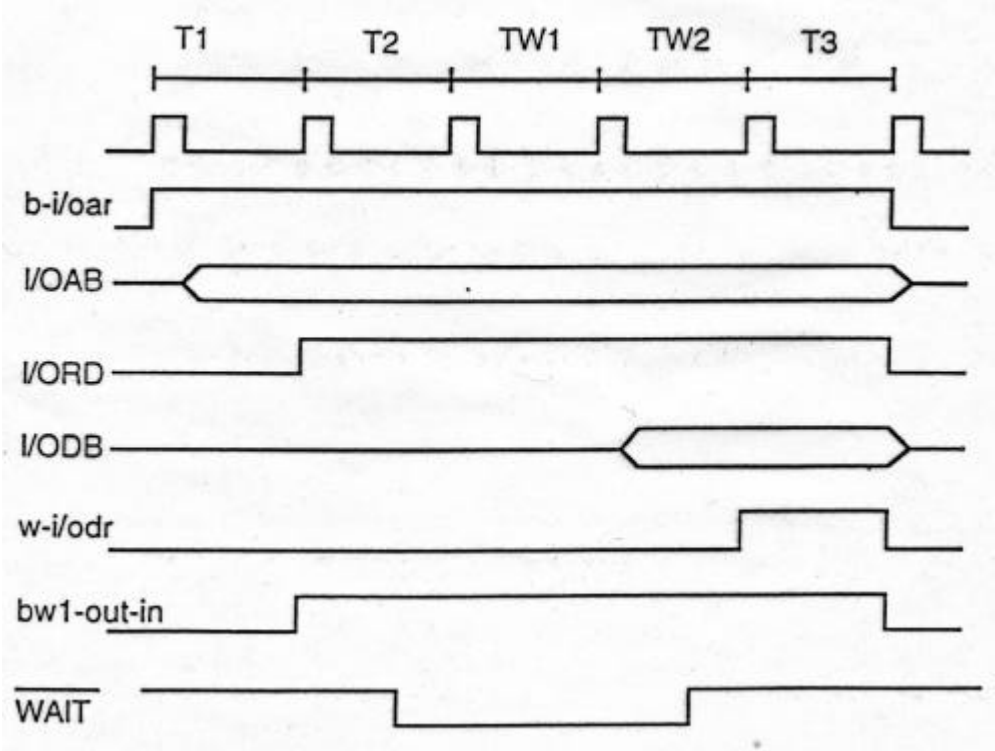
OUTPUT



Cicli di accesso
Interazione sincrona

- Dispositivo lento, ipotesi non soddisfatta.
- SCO esegue test su segnale WAIT, attivato da dispositivo, prima di completare l'interazione.
- In particolare viene effettuato il test al termine del 2° ciclo, questo viene ripetuto finche WAIT rimane basso (dispositivo dichiara di non essere pronto)

LETTURA

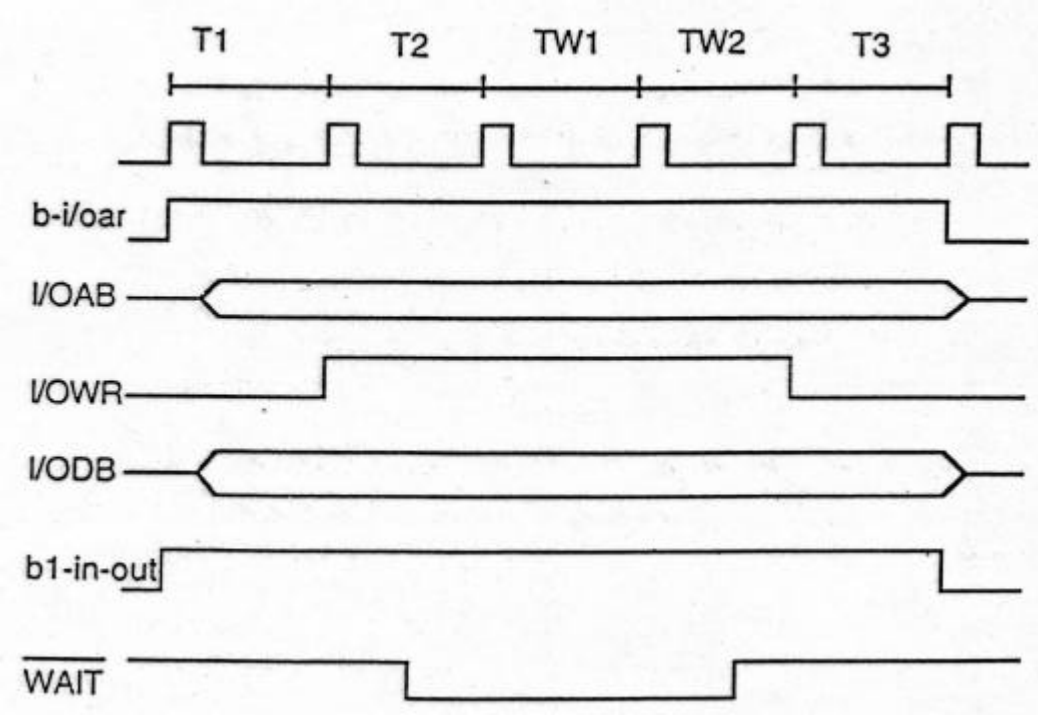


Abbiamo rete combinatoria del dispositivo che in base a segnali dal PD32 (I/OAB e IORD/IOWR) genera WAIT con un certo delay d, ipotizziamo che d < 1 ciclo CLK

Cicli di accesso
Interazione sincrona

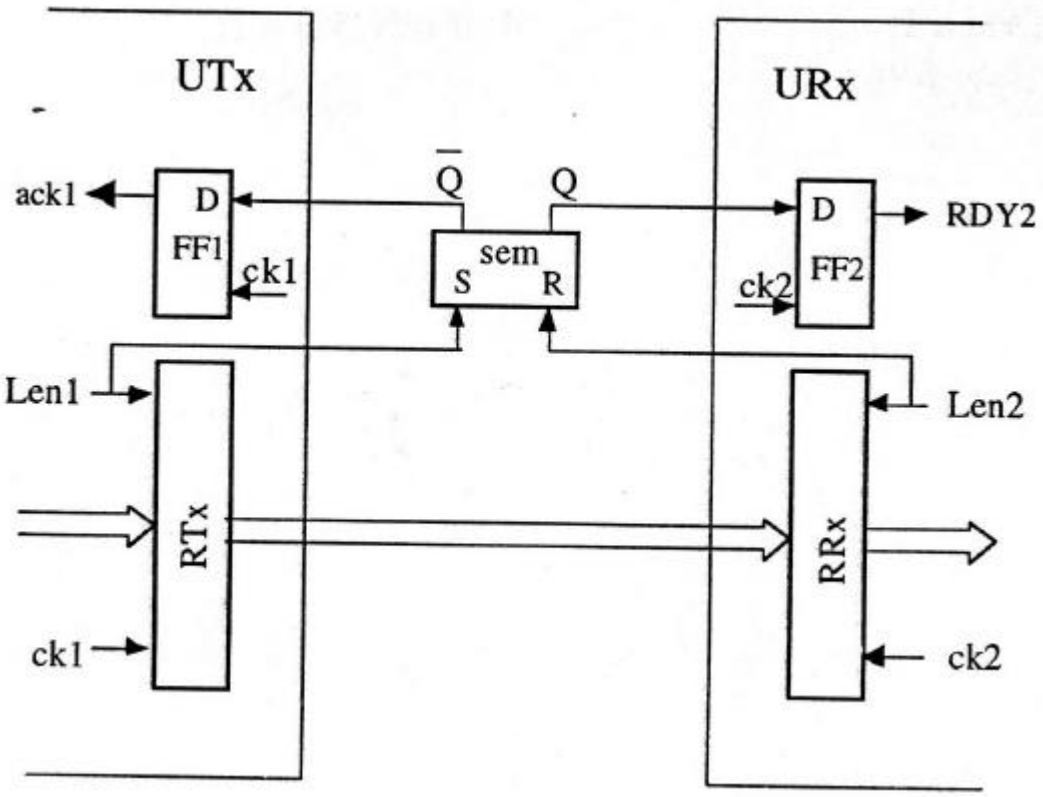
- Dispositivo lento, ipotesi non soddisfatta.
- SCO esegue test su segnale WAIT, attivato da dispositivo, prima di completare l'interazione.
- In particolare viene effettuato il test al termine del 2° ciclo, questo viene ripetuto finche WAIT rimane basso (dispositivo dichiara di non essere pronto)

SCRITTURA



Abbiamo rete combinatoria del dispositivo che in base a segnali dal PD32 (IOAB e IORD/IOWR) genera WAIT con un certo delay d, ipotizziamo che d < 1 anno

Sincronizzazione tra 2 unità



UTx	**URx**
1: Len1 ;dato->RTx	1: if RDY2=0 goto 1:
;set “Sem”	2: Len2 ;RTx->RRx
2: if ack1=0 goto 2:	;reset “Sem”
3: i+1->i	3: i+1->i
4: if i<>N goto 1:	4: if i<>N goto 1:

Prodotto tra due numeri relativi positivi

1. Specifiche

Realizzare una sub-routine di sistema che riceve parametri (indirizzo moltiplicatore, indirizzo moltiplicando, indirizzo risultato) e realizza funzione prodotto di 2 numeri con segno (complemento a 2) che sono pero' >=0.

Deve *segnalare gli overflow (trabocchi).*

I parametri (da passare) vengono posti sullo stack in quest'ordine:

Stack pointer (R7)	----->	PC
		Addr. Moltiplicando
		Addr. Moltiplicatore
		Addr. Risultato

2. Soluzione: classico algoritmo per addizioni successive.

Esempio:

				0	1	1	0	A
				0	0	1	1	B
			---	---	---	---	---	---
				0	1	1	0	
			0	1	1	0	-	
		0	0	0	0	-	-	
	0	0	0	0	-	-	-	
			---	---	---	---	---	---
0	0	0	1	0	0	1	0	

Overflow!

Algoritmo prodotto

```
/* A: moltiplicando */
/* B: moltiplicatore */
/* R: risultato */

R=0;
While (B<>0) {
    <shift di B a destra di 1 bit, con carry>;
    if (C<>0) {
          R=R+A;
          if (V==1)  <ritorno>;    /* overflow */
    }
     <shift di A a sinistra di 1 bit>;
     if (V==1)  <ritorno>;    /* overflow */
}   /*end while*/
<ritorno>    /* risultato corretto */
```

Subroutine prodotto

```
prodotto:   push r6             ; R6 punta allo stack
            movl r7, r6         ; del PD32
            addl #4, r6         ; punta il moltiplicando
            < push r2, r3, r4, r5 >
            movl (r6), r2       ; carico moltiplicando
            movl (r2), r3       ; in R3
            movl 4(r6), r2      ; carico moltiplicatore
            movl (r2), r4       ; in R4
            xorl r5, r5
ciclo:      clrc
            asrl #1, r4
            jnc continua
            addl r3, r5
            jv trabocco
continua:   cmpl #0, r4
            jz finemol
            asll #1, r3
            jv trabocco
            jmp ciclo
finemol:    movl 8(r6), r2      ; salvo risultato
            movl r5, r2         ; all'indirizzo specificato
trabocco:   < pop r5, r4, r3, r2, r6 >
            ret
```

PD32

Gestione delle Interruzioni

(VI)

Generazione delle interruzioni

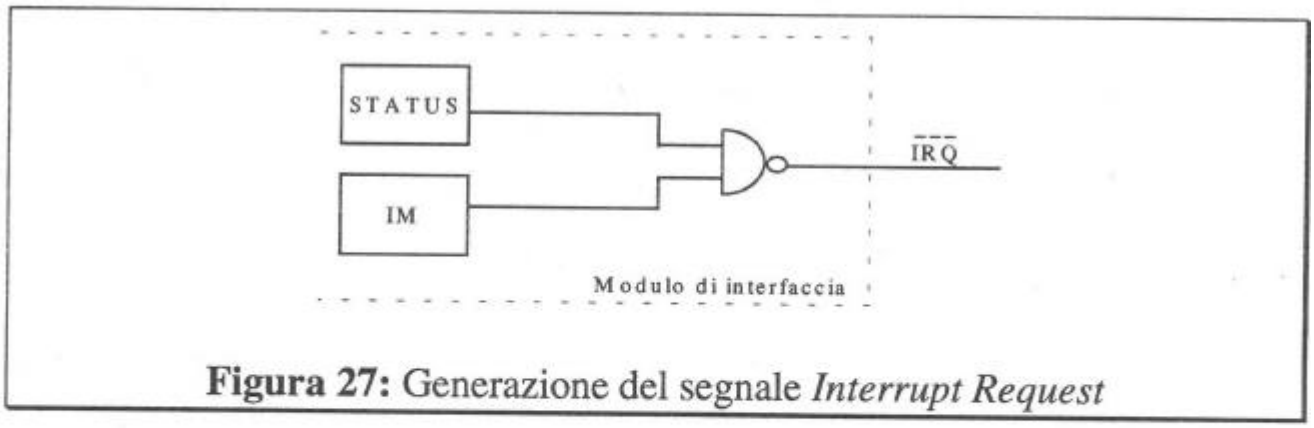


Figura 27: Generazione del segnale *Interrupt Request*

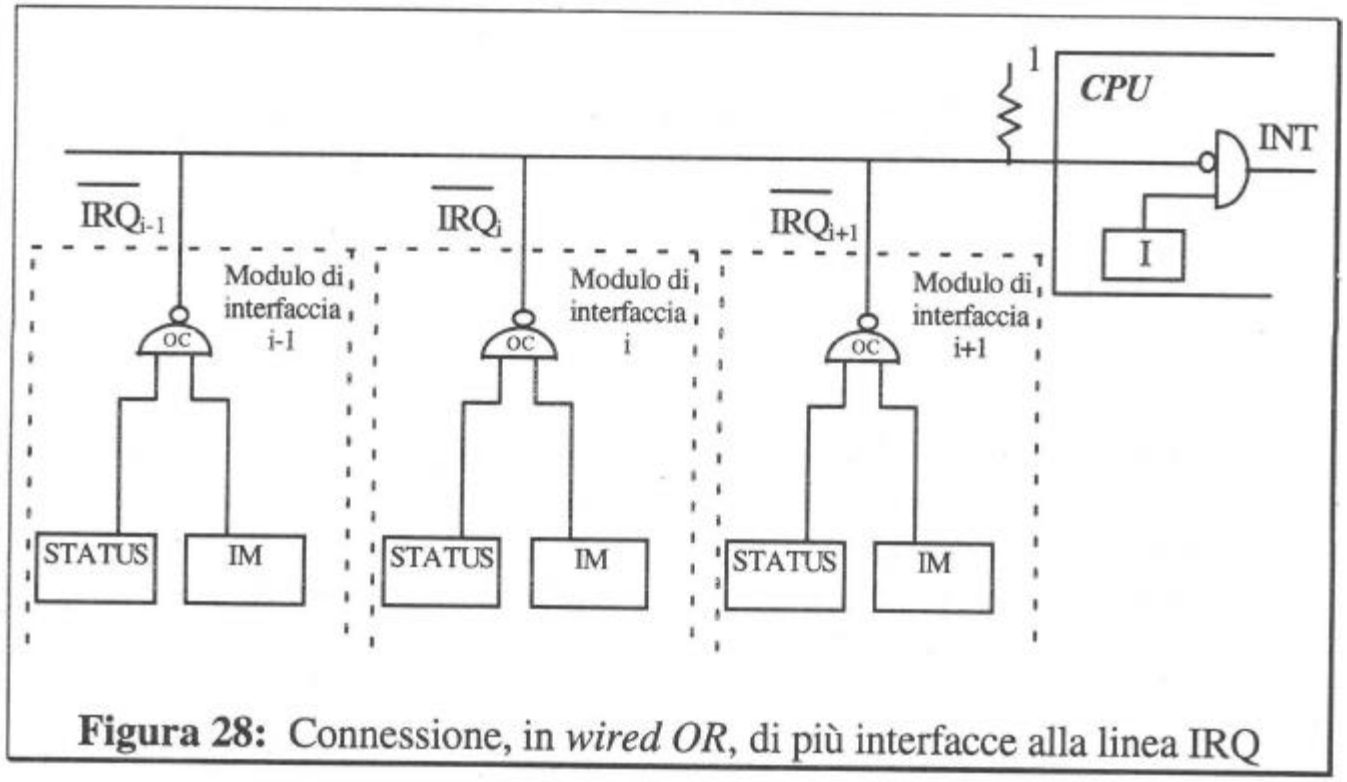


Figura 28: Connessione, in *wired OR*, di più interfacce alla linea IRQ

Gestione Interruzioni

- Fase di riconoscimento delle interruzioni
- Funzioni del sistema di interruzioni
  - Salvataggio del contesto minimo
  - Completamento del salvataggio
  - Ripristino del contesto
- Riconoscimento del dispositivo
  - Polling (via software)
  - Interruzioni vettorizzate (via hardware)
- Gerarchia di priorità

Interfacce di I/O complete di sotto-rete per il trattamento delle interruzioni

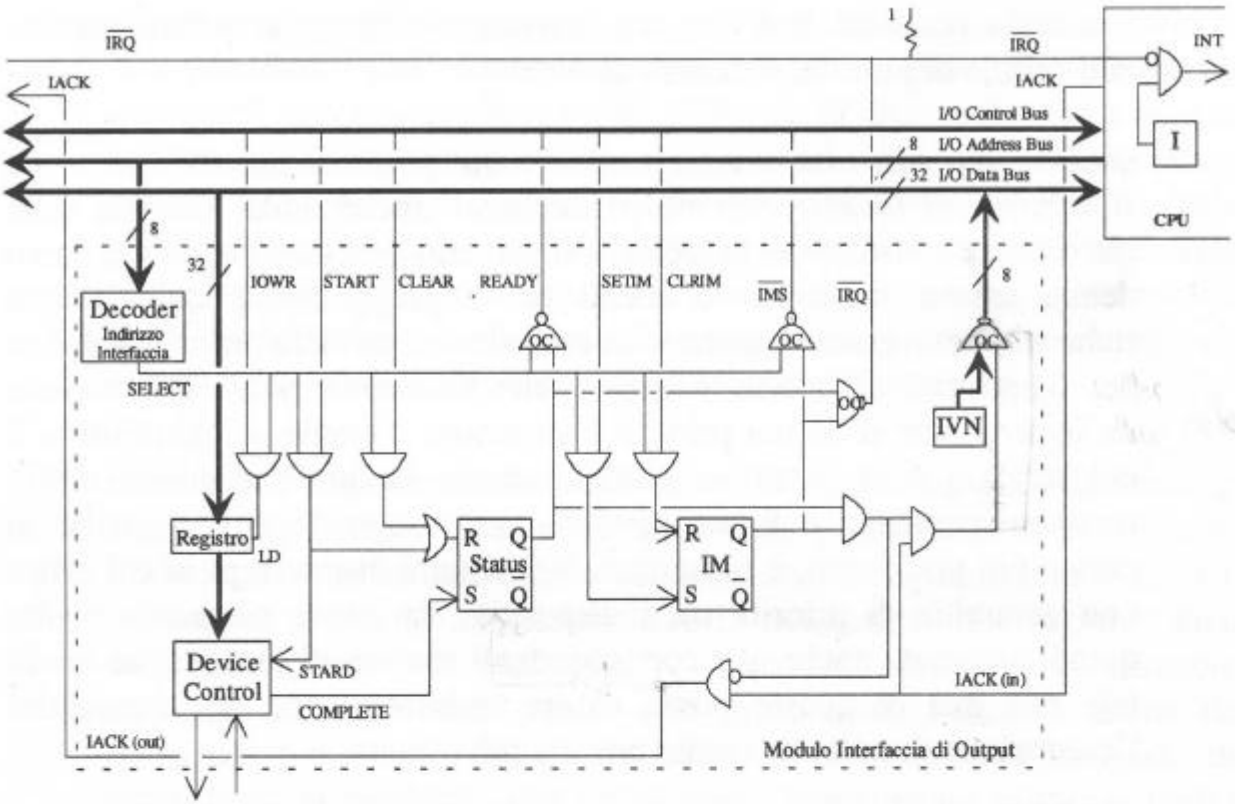


Figura 29: Interfaccia di output e sua connessione con il PD32

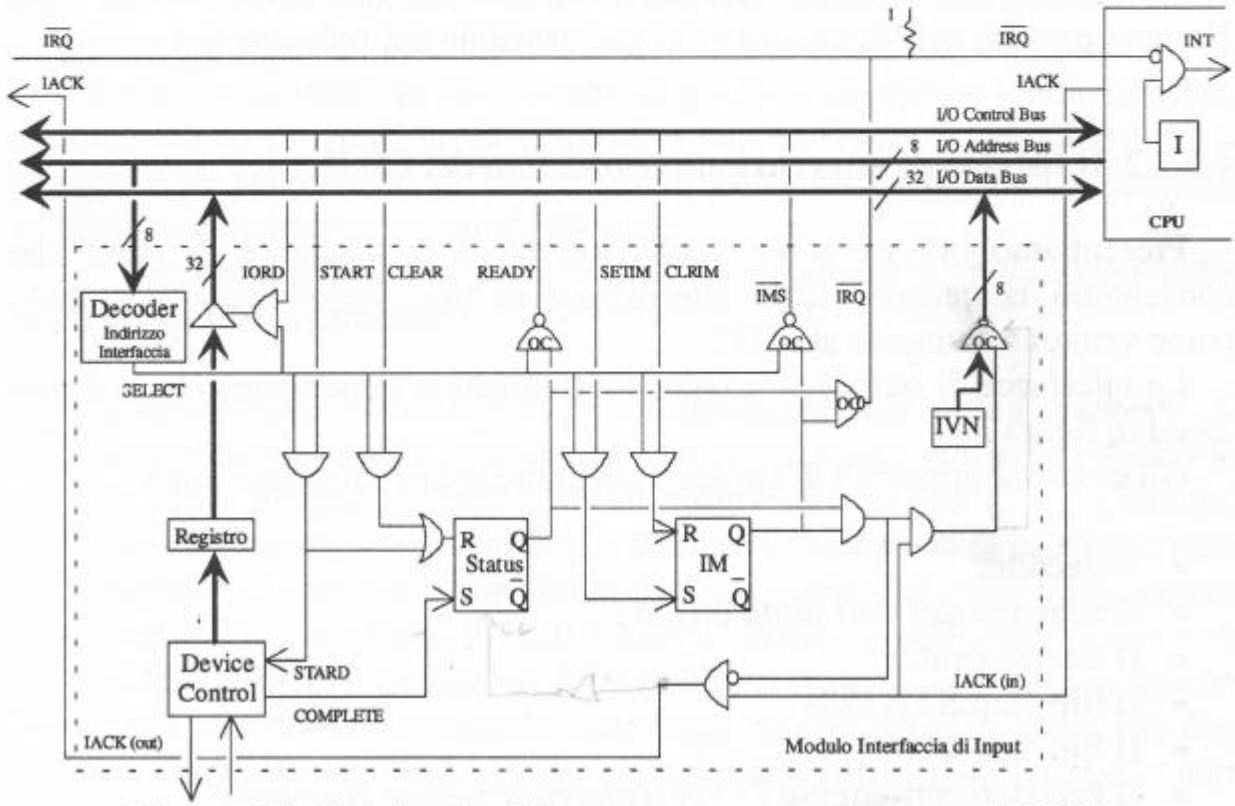


Figura 30: Interfaccia di input e sua connessione con il PD32

Connessione delle periferiche in Daisy Chain

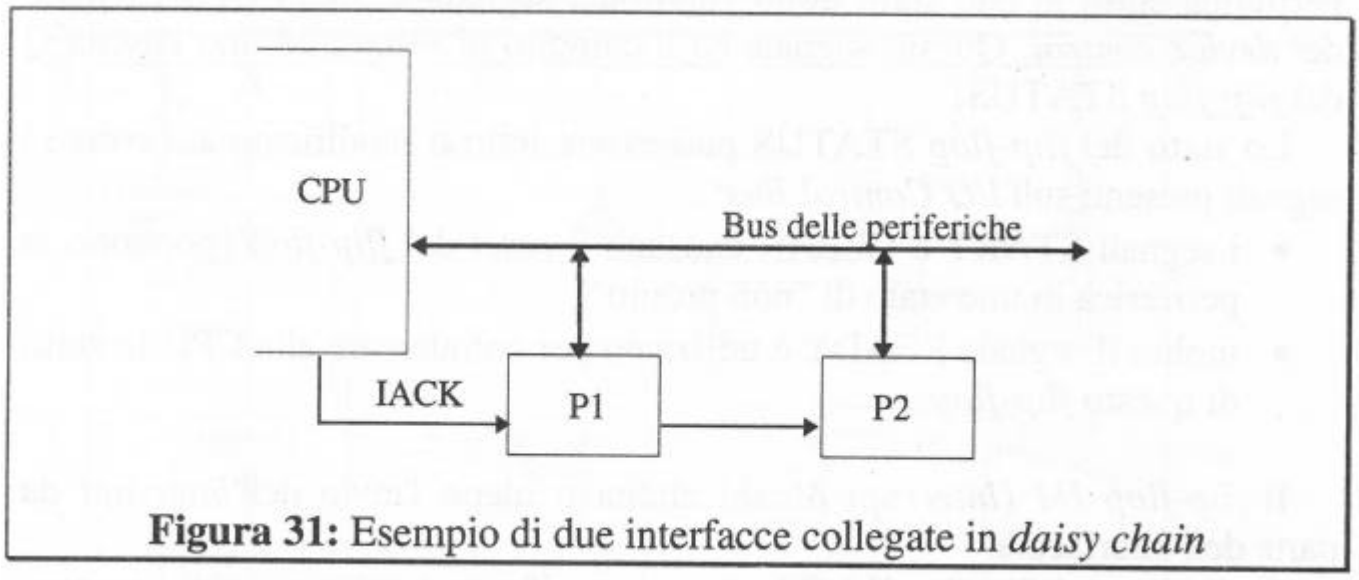
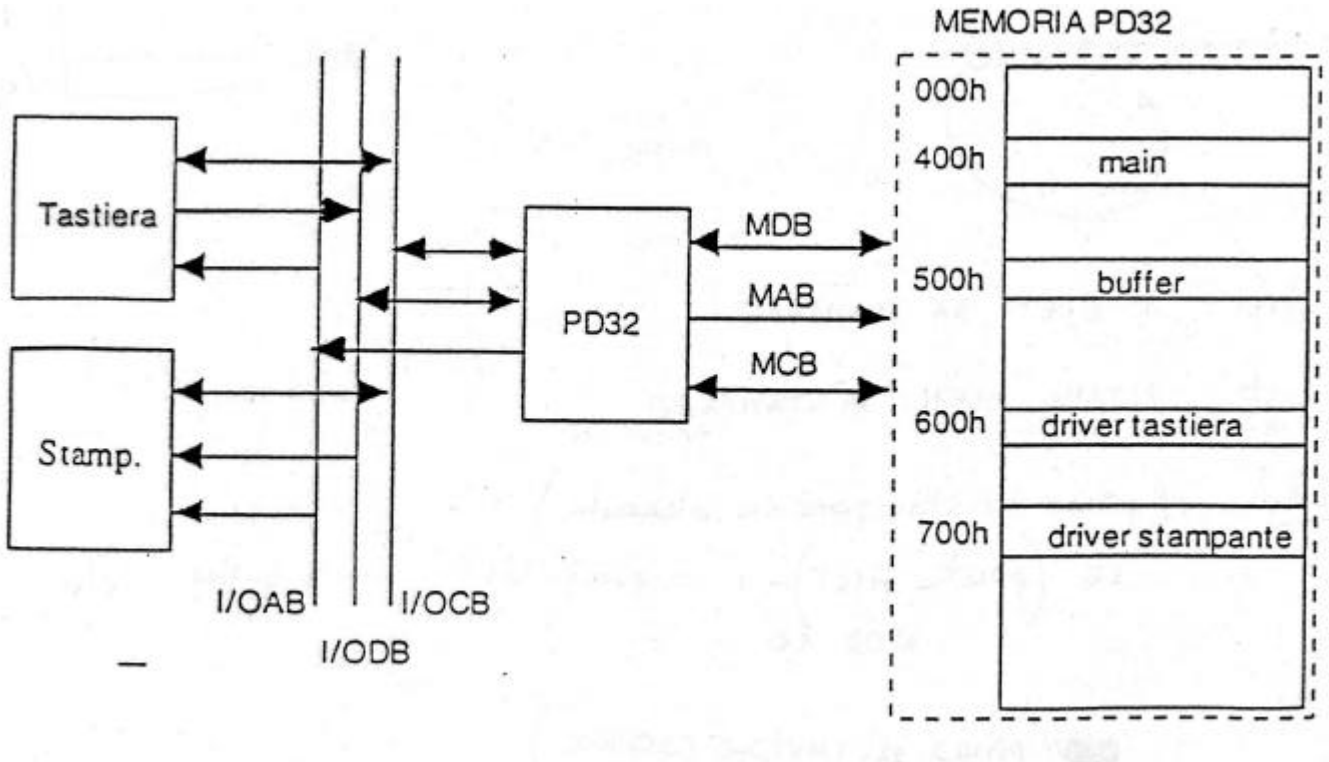


Figura 31: Esempio di due interfacce collegate in *daisy chain*

Esempio: Produttore-Consumatore

Specifiche:

- La tastiera invia un carattere alla volta, la stampante preleva un dato per volta e provvede alla stampa.
- Abbiamo un buffer di 10 caratteri
- Se il buffer è pieno la tastiera viene messa in attesa, idem per la stampante se il buffer è vuoto



Regole da implementare nei due driver per una corretta interazione tra i due dispositivi

Driver della tastiera

T1 –

Se il buffer è pieno e la tastiera cerca di inserire un nuovo dato

allora viene inibita la possibilità di fare nuove interruzioni da parte della tastiera

T2 –

Se la tastiera invia un dato e la stampante era stata bloccata (regola S1)

allora la stampante viene riabilitata ad effettuare nuove interruzioni

Driver della stampante

S1 –

Se il buffer è vuoto e la stampante cerca di leggere un dato

allora viene inibita, alla stampante, la possibilità di fare nuove interruzioni

S2 –

Se la stampante stampa un dato e la tastiera era stata bloccata (regola T1)

allora la tastiera viene riabilitata ad effettuare nuove interruzioni

```
; TASTIERA:      I/O=I, ind=30h, IVN=2
; STAMPANTE:  I/O=O, ind=56h, IVN=7

      org 400h        ;INIZIO PROGRAMMA

tast      equ 30h          ;indirizzo tastiera
stamp     equ 56h          ;indirizzo stampante
IVNtas    equ 2            ;IVN tastiera
IVNsta    equ 7            ;IVN stampante
Pfin      db 0             ;punt. primo dato buffer
Plast     db 0             ;punt. ultimo dato buffer
buffer    equ 500h         ;inizio buffer

      code

      seti             ;abilita interruzioni
      setim tast       ;abilita interruzioni tastiera
      setim stamp      ;abilita interruzioni stampante

main:     movl 0,r1        ;inizio progr. principale
          jmp main         ;fine progr. principale
```

```
;DRIVER TASTIERA
    driver IVNtas, 600h        ;inizia dall'ind. 600h

    push r0
    push r1
    movb Pfin, r1
    movb Plast, r0
    cmpb 0, r0                 ; aggiornamento modulo
    jnz label1                 ; 10 del puntatore
    movb #10, r0               ; punt-last
label1:  subb #1, r0           ; --------------------
         cmpb r1, r0
         jz BlocTas            ; buffer pieno,regola T1
scrivi:  inb tast, buffer(r0)
         movb r0, Plast
         start tast            ; abilito produzione prox dato
    jnim stamp, SblocSta       ; sblocca stampante se era
                               ; bloccata da S1; regola T2
fineTas: pop r1
         pop r0
         rti
BlocTas: clrim tast
         jmp fineTas
SblocSta: setim stamp
         jmp fineTas
```

```
;DRIVER STAMPANTE
            driver IVNsta,700h ;inizia dall'ind. 700h

            push r0
            push r1
            movb Pfin, r1
            movb Plast, r0
            cmpb r1, r0
            jz BlocSta          ;buffer vuoto regola T1
leggi:      outb buffer(r1), stamp
            cmpb 0, r1          ; aggiornamento modulo
            jnz label2          ; 10 del puntatore
            movb #10, r1        ; punt-first
label2:     subb #1, r1         ; --------------------
            movb r1, Pfin
            start stamp         ; abilito consumo dato
            jnim tast, SblocTas ; sblocca tastiera se era
                                ; bloccata da T1; regola S2
fineSta:    pop r1
            pop r0
            rti
BlocSta:    clrim stamp
            jmp fineSta
SblocTas:setim tast
            jmp fineSta
;///----------fine driver  ------------///
     end
```