GESTIONE DEGLI EVENTI

Gestione degli Eventi

a controllo di programma
POLLING
GESTITO DAL MASTER

ad interruzioni
gestione di eventi asincroni
disaccoppiamento delle funzioni

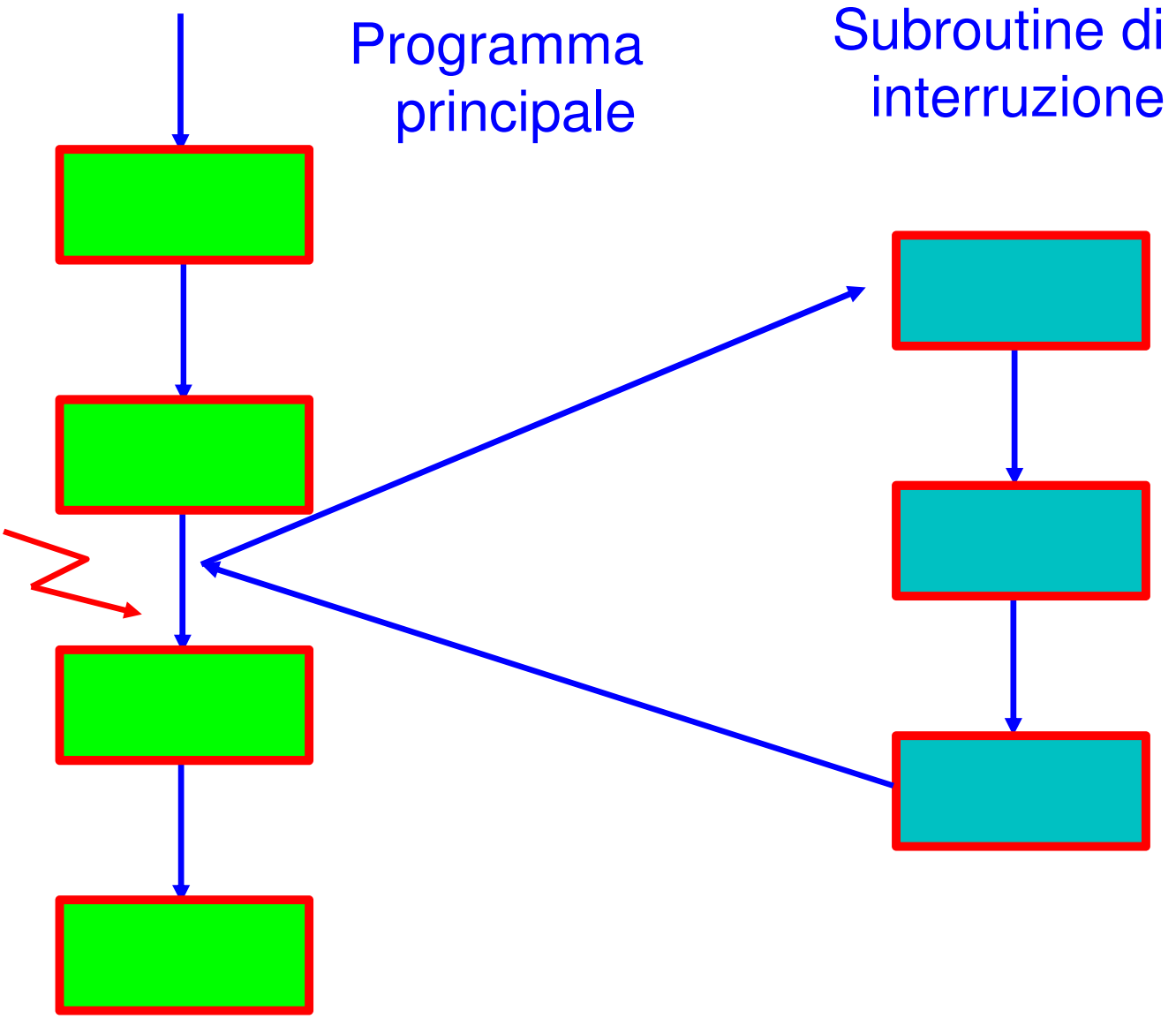
INTERRUZIONE
sospensione forzata del processo di esecuzione e trasferimento di controllo ad una " routine di servizio"
che soddisfa le richieste dell'evento che ha provocato l'interruzione, al termine della quale il controllo viene restituito al processo sospeso

INTERRUZIONI
"veri" interrupt, asincroni con l'esecuzione del programma:
- HARDWARE ALTA PRIORITA' (non mascherabili)
- HARDWARE BASSA PRIORITA' (mascherabili)

semplici istruzioni o side effect di istruzioni:
- SOFTWARE NEL PROGRAMMA
- SOFTWARE INDIRETTE (STATO DELLA CPU: trap)

INTERRUZIONI

- Sistemi a controllo di programma: cattivo sfruttamento della CPU, lentezza, impossibilità a rispondere con prontezza agli stimoli esterni



INTERRUZIONI

- Il segnale di interruzione è riconosciuto sul livello
- Istruzioni di abilitazione e disabilitazione delle interruzioni hardware mascherabili (bit GIEL)
- Interruzioni automaticamente disabilitate al RESET (reset configuration: GIE=GIEL=GIEH=0)
- Le interruzioni possono essere multiple
- Possono essere gestite dal micro utilizzando le sole sorgenti interne di interrupt, oppure con un interrupt controller esterno
- Gestione dell'interruzione da parte del micro in associazione all'interrupt controller esterno 8259: all'atto del riconoscimento di una interruzione hardware (sempre al termine di una istruzione e mai durante la sua esecuzione) si deve generare per due volte consecutive il segnale INTA*.
  Al secondo INTA* deve essere generato sul bus dati basso dall'esterno un byte che viene letto dal microprocessore
- Il byte letto viene utilizzato per puntare ad una tabella di indirizzi di procedure di servizio alle interruzioni che vengono interpretate come puntatori a 4 bytes, che contengono il codice della funzione di risposta

INTERRUZIONI

3 INTERRUZIONI

Interruzioni riservate:

- **err. di divisione** – divisione per 0 (eccezione di cpu)

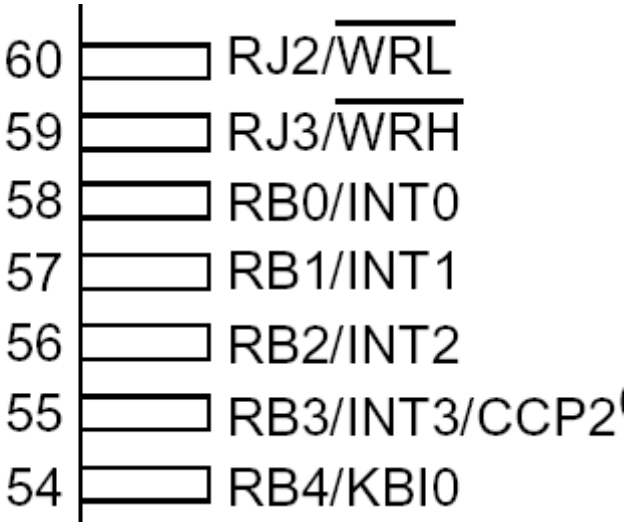
- **overflow** (eccezione di cpu)

altre interruzioni predefinite nelle cpu:
superamento di limiti del segmento (protezione su PC overflow)
protezione
codice non valido (viene eseguita una NOP)
watch dog interno
stack overflow/underflow
...

TABELLA DELLE INTERRUZIONI

Oltre a quelle hardware, se vogliamo generarle via software e decodificarle con un controller esterno, dobbiamo costruire una tabella di procedure da eseguire all'occorrenza di ciascuna causa esterna di interrupt. Tutto deve essere gestito dalla funzione di interrupt che, alla chiamata generata mediante pin di interrupt RBx, interroga il controller esterno, ricevendo da quest'ultimo il codice della interruzione che ha generato la chiamata; il secondo passo sarà l'esecuzione della funzione di servizio dell'interrupt corrispondente all'evento esterno occorso. Tale tabella sarà una tabella di indirizzi, diversa dalla "vector table"

Interrupt Interni del PIC18F8x20



Il micro è dotato di 4 sorgenti di interrupt esterne che sono interfacciabili ad altrettanti PIN della PORTB: RB0, RB1,RB2 e RB3.

Ciascun Interrupt è abilitato da un bit di abilitazione in un registro, è dotato di un bit che ne sancisce la priorità e l'occorrenza di un interrupt è determinata dal livello logico di un terzo bit; ad esempio per RB1 si ha:

INT1IE = abilitazione

INT1IP = priorità

INT1IF = Flag di interrupt (se =1 c'è richiesta di interrupt)

Possono essere attribuite due sole priorità: High e Low; ciascun interrupt High può interrompere un qualsiasi Low in qualsiasi momento.

Interrupt Interni

La vector table è il vettore degli indirizzi delle funzioni che vengono automaticamente invocate al riconoscimento di un interrupt. Nonostante il micro abbia molte possibili sorgenti di interrupt, vi sono sole due procedure previste da chiamare, che corrispondono a due indirizzi di Flash: 00008h per gli interrupt High priority e 00018h per gli interrupt low priority.

Al riconoscimento di un interrupt quindi l'indirizzo di ritorno del PC viene salvato sullo stack e il PC viene caricato con l'address della funzione da eseguire (8h o 18h in base al tipo di interruzione riconosciuta).

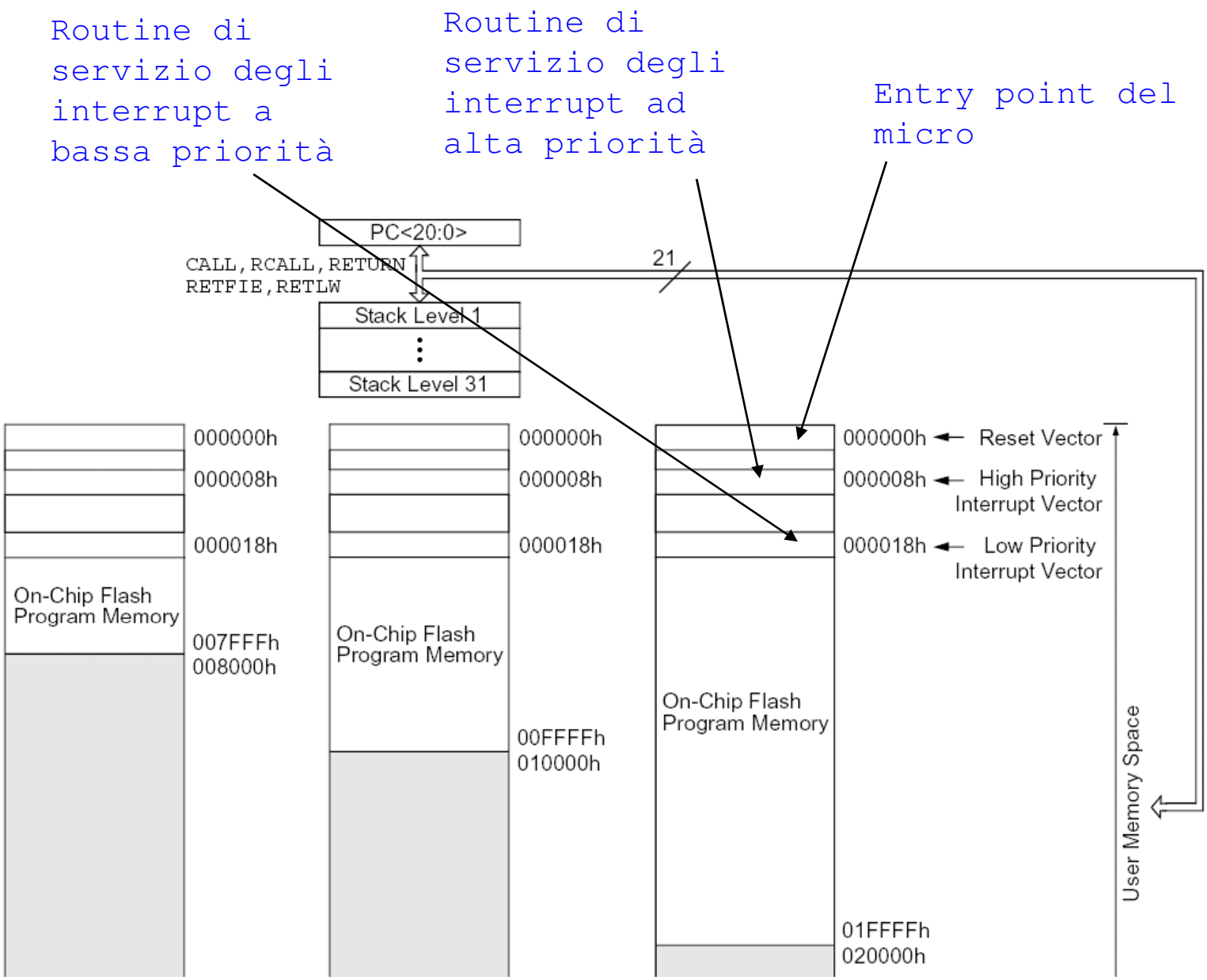
Dato che le cause delle interruzioni possono essere molteplici, esse vengono riconosciute in base al valore logico del pin di "Flag" che può essere controllato via software.

Le priorità vengono abilitate settando il bit IPEN in RCON register (1= priorità abilitate).

Gli altri bit danno informazioni sulle cause dell'ultimo reset occorso (vedere par 4.14 datasheet micro).

R/W-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0
IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

Allocazione dei puntatori alle funzioni di servizio degli interrupt



Sono indirizzi e quindi sono formati da almeno 3 byte di indirizzo e contengono il puntatore a funzione che indica l'address delle funzioni di servizio degli interrupt generate dal programmatore software e "linkate" nella flash assieme al resto del codice utente.

Interrupt Interni

```
void interrupt IntHighMgm( void )
{
 do{
    if (INT0IE && INT0IF)//controllo Flag ed enable
    {
       FunzAssociataINT0();
    }
    if (INT1IE && INT1IF) //controllo Flag ed enable
    {
       FunzAssociataINT1();
    }
    if (INT2IE && INT2IF) //controllo Flag ed enable
    {
       FunzAssociataINT2();
    }
    if (INT3IE && INT3IF) //controllo Flag ed enable
    {
       FunzAssociataINT3();
    }
    if (TMR0IE && TMR0IF) //controllo Flag ed enable
    {
       FunzAssociataINT0();
    }
 }while(INT0IF || INT1IF || INT2IF || INT3IF);
}
```

In tal modo se si verifica una richiesta di un interrupt mentre se ne sta servendo un altro, , all'uscita della funzione si controlla lo stato di tutti i flag e non si perdono gli altri interrupt.

Funzione di servizio dell'interrupt

```
void FunzAssociataINTx( void )
{
    INxIE=0; //disable external interrupt x
    INTxIF=0; //clear interrupt Flag
    // Enter Your code here

    INTXIE=1; //re-enable external interrupt x
}
```

Il flag di interrupt deve essere resettato via software, mentre l'interrupt deve essere disabilitato durante l'esecuzione della intera funzione di interrupt.

Dato che gli interrupt di bassa priorità possono essere interrotti da quelli di alta priorità, se non si desidarno interruzioni si devono disabilitare tutti gli interrupt durante la routine di servizio dell'interrupt low priority:

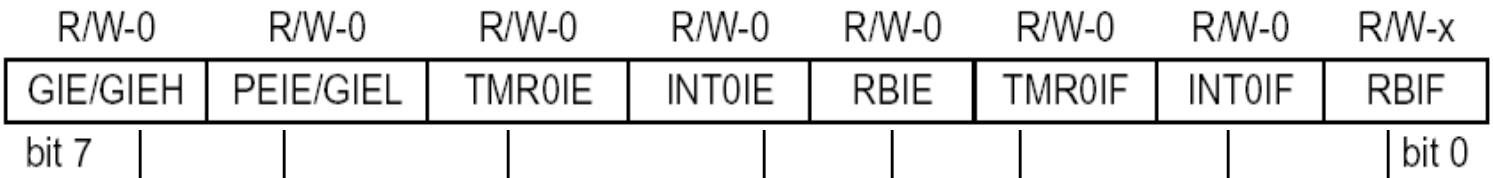
```
GIEH=0; //High priority interrupt disable
GIEL=0; //Low priority interrupt disable
```

Per poi riabilitarli alla fine della procedura,

```
GIEH=1; //High priority interrupt enable
GIEL=1; //Low priority interrupt enable
```

ma è una funzionalità da utilizzare con molta attenzione……

Registri interno associati: INTCON



GIE=Global Interrupt enable bit (1=Enable)

TMR0IE=Timer 0 overflow interrupt Enable bit (1=Enable)

INT0IF= External INT0 Flag bit (1=interrupt occurred)

PEIE/GIEL=Peripheral Interrupt Anble bit (Low priority interrupt enable bit) (1= enable)

INT0IE= External INT0 enable bit

RBIE=RB port change Interrupt Enable bit (1=Enable)

TMR0IF=Timer 0 overflow interrupt Flag bit (1=interrupt occurred)

RBIF=RB port change Interrupt Flag bit (1=int. occurred)

RB Interrupt sta ad indicare che viene generato un interrupt se almeno uno dei bit della porta B: RB4, RB5, RB6, RB7 ha cambiato stato (interrupt su livello).

Registri interno associati: INTCON2

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP
bit 7							bit 0

RBPU*=PORTB Pull-up enable bit (per open collector interrupt source (ad esempio diagnostica di componenti esterni - vedere BTSXXX della infineon) (0=Enable)
INTEDG0=External Interrupt 0 edge select bit (1= rising edge)
INTEDG1=External Interrupt 1 edge select bit (1= rising edge)

INTEDG2=External Interrupt 2 edge select bit (1= rising edge)

INTEDG3=External Interrupt 3 edge select bit (1= rising edge)

TMR0IP=Timer 0 overflow interrupt Priority bit (1=High priority)
INT3IP=External INT3 priority bit (1=High priority)
RBIP=RB port change Interrupt priority bit (1=High priority)

RB Interrupt sta ad indicare che viene generato un interrupt se almeno uno dei bit della porta B: RB4, RB5, RB6, RB7 ha cambiato stato (interrupt su livello).

Registri interno associati: INTCON3

R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF
bit 7							bit 0

INT2IP= External INT2 priority bit (1=High priority)
INT1IP= External INT1 priority bit (1=High priority)
INT3IE=External Interrupt 3 enable bit (1= enable int.)
INT2IE=External Interrupt 2 enable bit (1= enable int.)
INT1IE=External Interrupt 1 enable bit (1= enable int.)
INT3IF=External Interrupt 3 Flag bit (1= int. occurred)
INT2IF= External Interrupt 2 Flag bit (1= int. occurred)
INT1IF= External Interrupt 1 Flag bit (1= int. occurred)

N.B. Tutti gli interrupt flag devono essere resettati via software

Interrupt collegati alle funzionalità delle periferiche

Vi sono altri 9 registri che permettono la attivazione degli interrupt collegati alle funzionalità delle periferiche integrate del micro (seriali, timer, moduli CCP - capture/compare/PWM, PSP - parallel slave port, AD converter, Comparatore, Bus collision (I2C mode)……

Di ogni sorgente di interrupt, analogamente a quanto già visto, si possono settare l'abilitazione e la priorità, mentre lo stato è osservabile mediante il test del Flag bit (che deve comunque essere resettato via software utente nella procedura di servizio dell'interrupt).

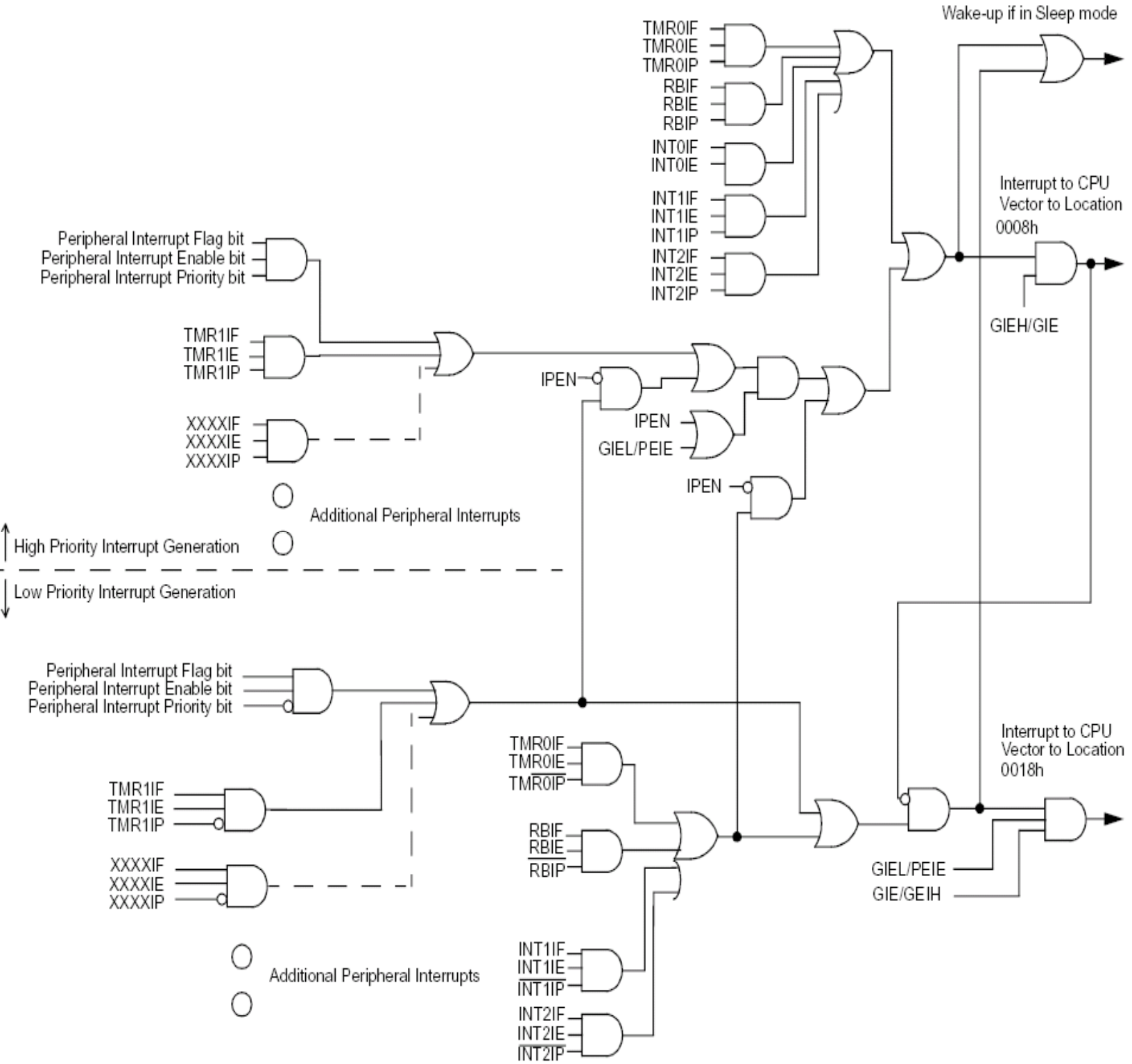
Di seguito vengono elencati i registri che sono nel data sheet del micro al capitolo 9 "Interrupt".

PIR1, PIR2, PIR3 = Peripheral Interrupt Flag bits
PIE1, PIE2, PIE3 = Peripheral Interrupt Enable bits
IPR1, IPR2, IPR3 = Peripheral Interrupt priority bits

Con significati analoghi ai bit analizzati finora.

N.B. INT0IF è SEMPRE ad alta priorità, non è quindi possibile regolarne la priorità ed il relativo bit non esiste.

Struttura delle interruzioni

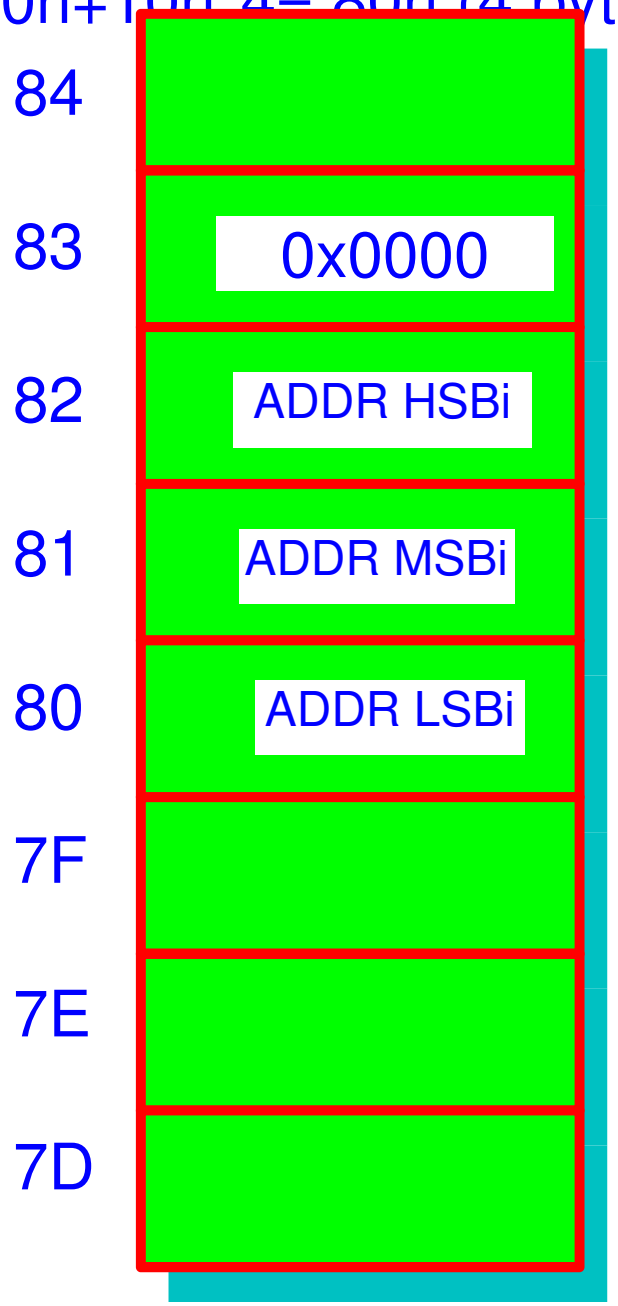


Interrupt, periferiche interne ed esterne al micro

Il micro in oggetto ha una struttura orientata all'utilizzo di periferiche interne ad esso.
Nel caso in cui si desiderino utilizzare risorse esterne (periferiche, componenti, dispositivi di IO), e i 4 PIN di interrupt non siano sufficienti, allora è possibile utilizzare un Interrupt controller esterno, ovvero una periferica in grado di gestire diverse sorgenti di interrupt, e in grado di mantenere traccia delle sorgenti che hanno attivato le diverse richieste, gestendo un complesso sistema di priorità.
Attraverso un protocollo di scambio dati che sarà analizzato, il dispositivo comunica al micro l'indice della sorgente che ha generato al richiesta. Sono possibili fino a 64 sorgenti esterne.
E' ovvio che alle 64 sorgenti esterne di interrupt corrispondono altrettante routine di servizio per soddisfare le richieste, e quindi saranno generati fino a 64 puntatori a funzione di 4 byte ciascuno (almeno 3 byte ciascuno). Il vettore composto da tutti questi indirizzi è detto "vettore delle interruzioni" o "vector table" e deve essere gestito via software dall'utente, una volta associato un ingresso di interrupt hardware del micro alla comunicazione tra micro e dispositivo esterno di controllo delle interruzioni. E' infatti necessaria una via di comunicazione prioritaria tra micro e dispositivo esterno per garantire bassi tempi di latenza della richiesta.

INTERRUZIONI

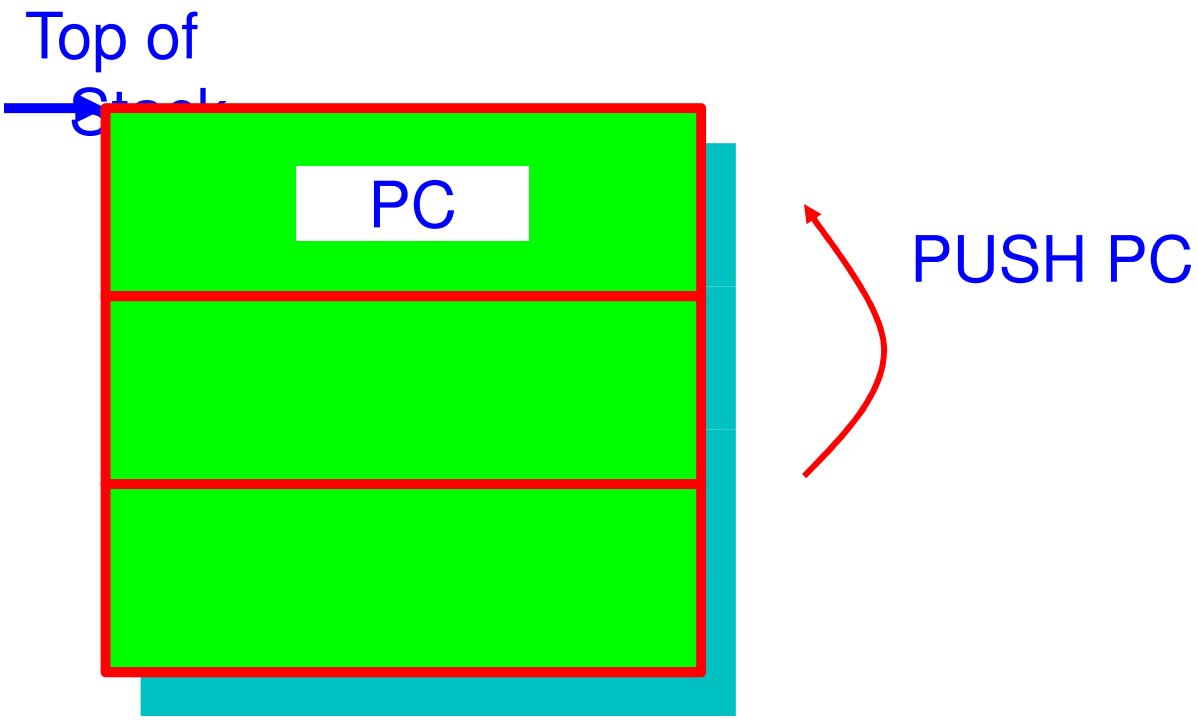
Supponiamo che l'INT TYPE generato dal dispositivo esterno sia 00010000b = 10h, e supponiamo che il vettore delle interruzioni sia allocato dall'indirizzo 40h.
40h+10h*4= 80h (4 byte per ogni indirizzo)



Interrupt vector
posizione n. 16
Indirizzo di 4 byte di cui
uno non significativo
Interrupt i-esimo

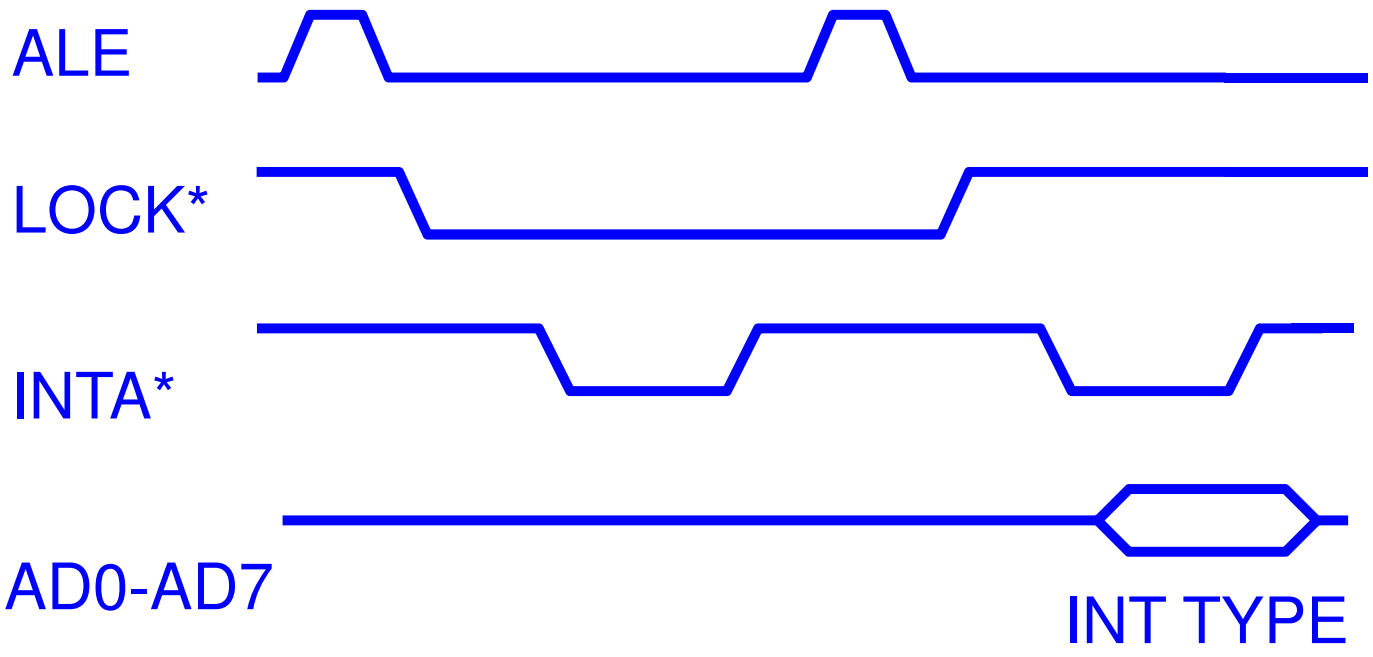
Ad esempio: Indirizzi da 64 a 64 + 64*4= 320 byte cioè da 40h a 180h riservati per la vector Table nel codice utente

Salvataggio del contesto durante gli interrupt



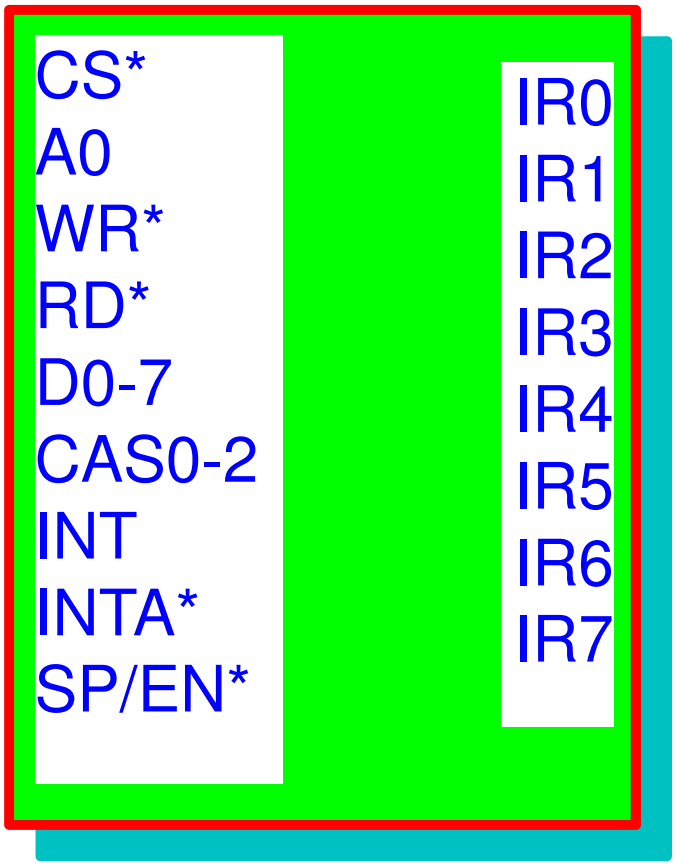
- All'atto del riconoscimento di una interruzione si hanno automaticamente le operazioni di PUSH PC nello stack, PUSH WREG, PUSH Status e PUSH BSR nel “fast return stack”.
- All’atto di ritorno dalla funzione di servizio dell’interrupt, i registri vengono ripristinati.
- Vi sono opzioni nei compilatori C per aggiungere a questa lista anche altri registri che saranno salvati in uno stack software generato dal compilatore
- I fast register non possono essere utilizzati contemporaneamente da High e low priority interrupt perché verrebbero sovrascritti.
- Interruzioni software: Molto usate nei sistemi operativi, la loro funzionalità è gestita via software dal sistema operativo stesso.

INTERRUZIONI

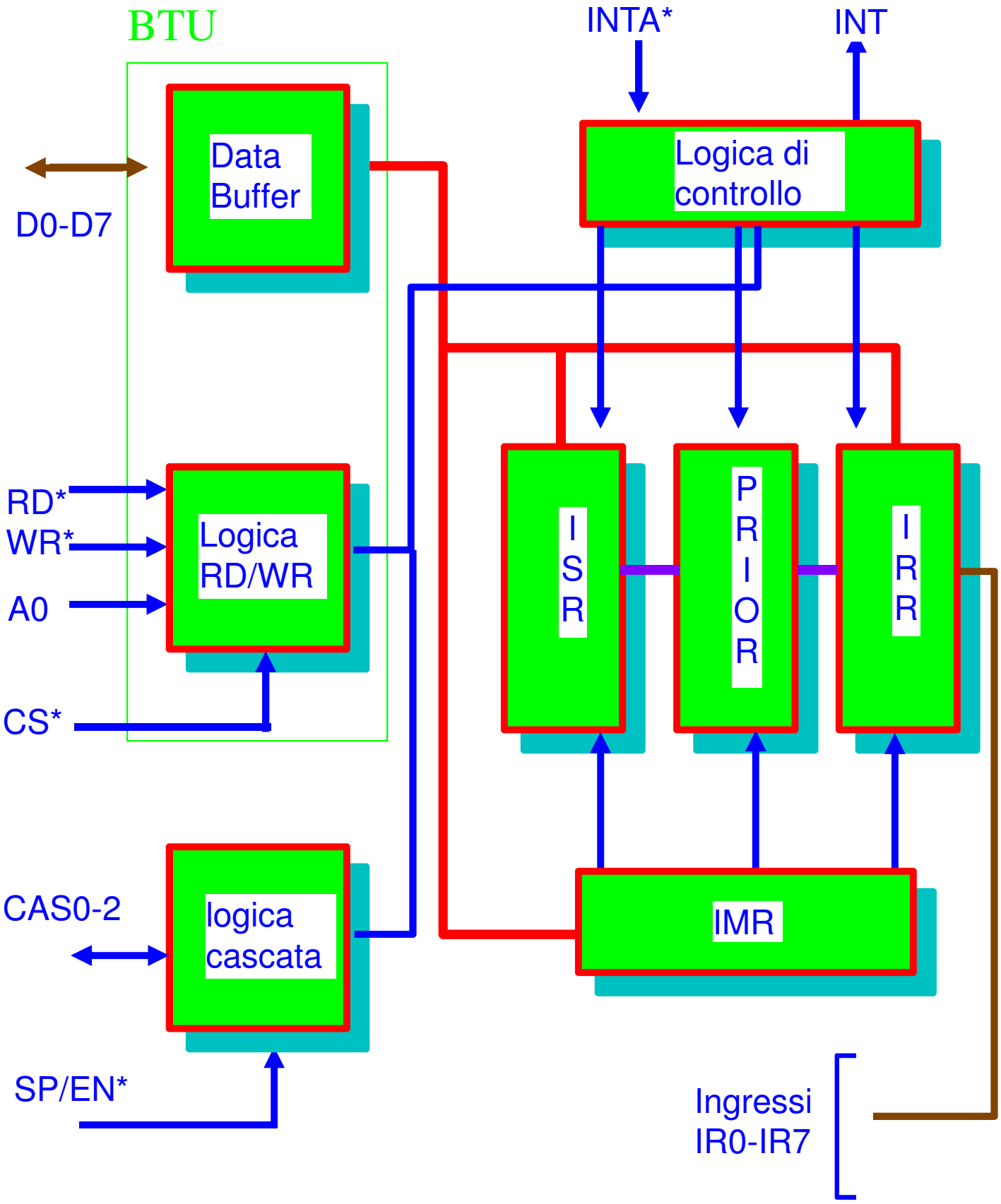


- Normale ciclo di bus in cui il segnale INTA* sostituisce il segnale di READ*
- I segnali di WAIT* sono riconosciuti anche durante il ciclo di interruzione (TW)
- Gli indirizzi emessi dal microprocessore sono privi di significato: i segnali ALE sono generati solo per indicare l'inizio del relativo ciclo di bus
- Normalmente il primo ciclo di INTA* serve a risolvere i conflitti e a congelare la situazione
- Il LOCK* serve spesso a distinguere tra il primo e il secondo ciclo di INTA*

INTERRUPT CONTROLLER 82(C)59



- Gestisce 8 interruzioni da solo e fino a 64 se connesso in cascata
- Occupa 2 locazioni di indirizzo (4 nel micro con divisione tra banco pari e banco dispari senza hardware di collegamento tra le due parti del bus dati): un solo PIN di address (A0)
- Riconosce le interruzioni esterne a livello o sul fronte positivo (dipende dalla programmazione)
- Durante il ciclo di riconoscimento della interruzione il segnale INTA* ingloba CS* e RD*
- Permette la modifica di: priorità, mascheratura, polling scelta dell'EOI e delle priorità, etc.



IRR: Interrupt Request Register
ISR: In-Service Register
IMR: Interrupt Mask Register
SP/EN*: Slave Program/Enable Buffer

INTERRUPT CON 8259

ciclo di interrupt con 8259 (pseudo-codice):

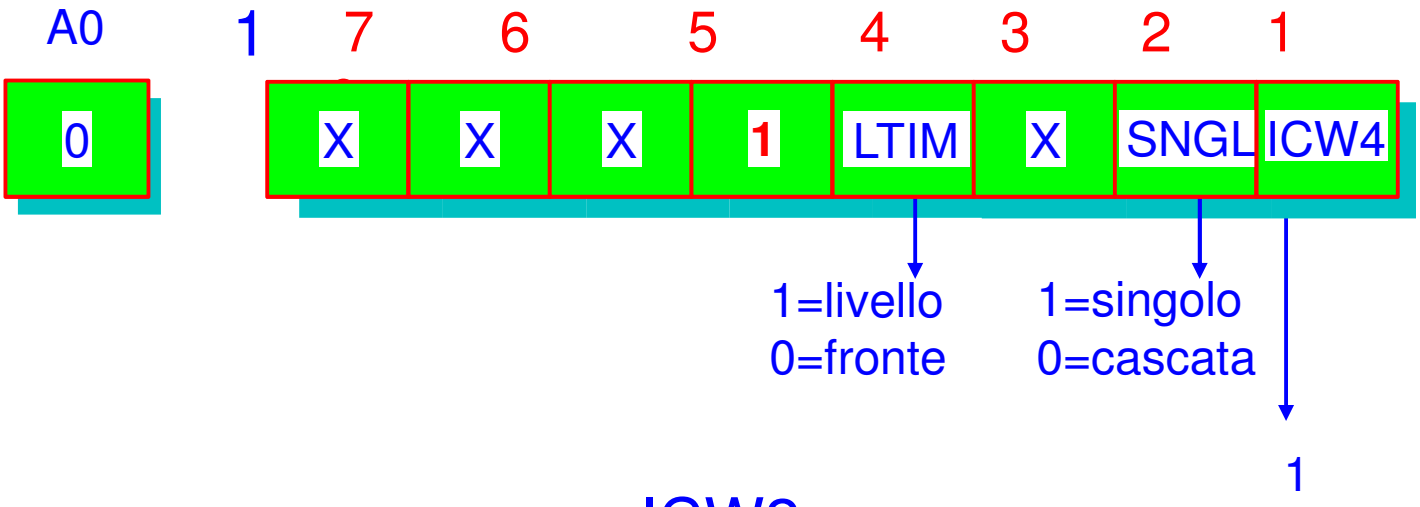
```
if (IRi )
  {set IRRi
   if (IMRi=0) {gestione priorità
                send intr
                receive inta - gestione priorità ( set iSRi)
                 if (edge) reset IRRi
                 receive inta
                 send INT TYPE ( base+i)
                 if AEOI  reset ISR }
   }
```

Programmazione dell' 8259

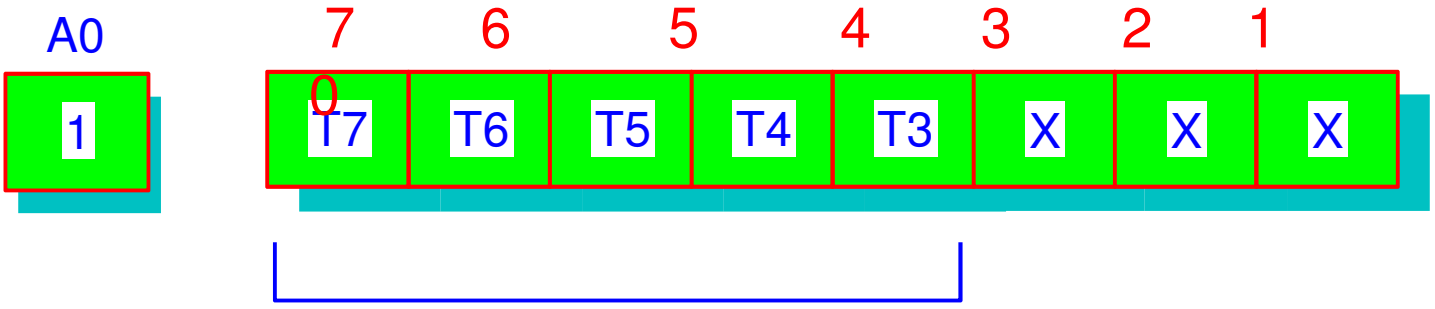
- programmazione **ICW1** (**livelli**, **single**, **ICW4**)
- programmazione **ICW2** (**type**)
- if **cascade** programmazione **ICW3**
- if **ICW4** programmazione ICW4 (**sfnm, buff, aeoi**)
- programmazione di eventuali slave
- programmazione delle parole di stato operative
  - ocw1 (**mask**)
  - ocw2 (**eoi**, priorità)
  - ocw3 (special mask mode …)

PAROLE DI CONTROLLO 8259

ICW1=Initialisation Command Word



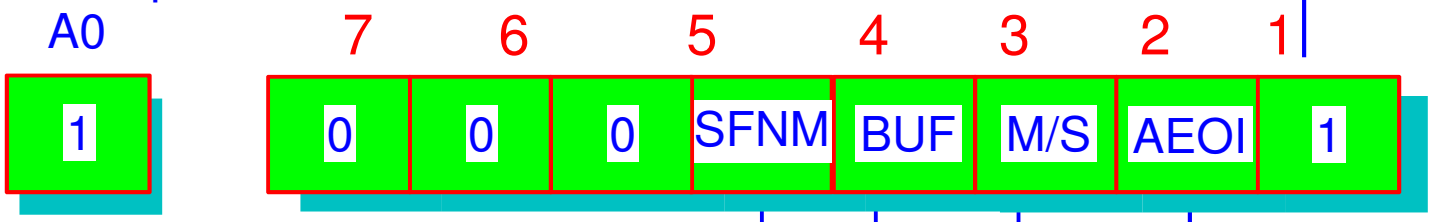
ICW2



5 bit più significativi del vettore di interruzione:
Contribuisce a costruire l'indice per il puntatore

ICW4

Compatibilità con vecchi micro

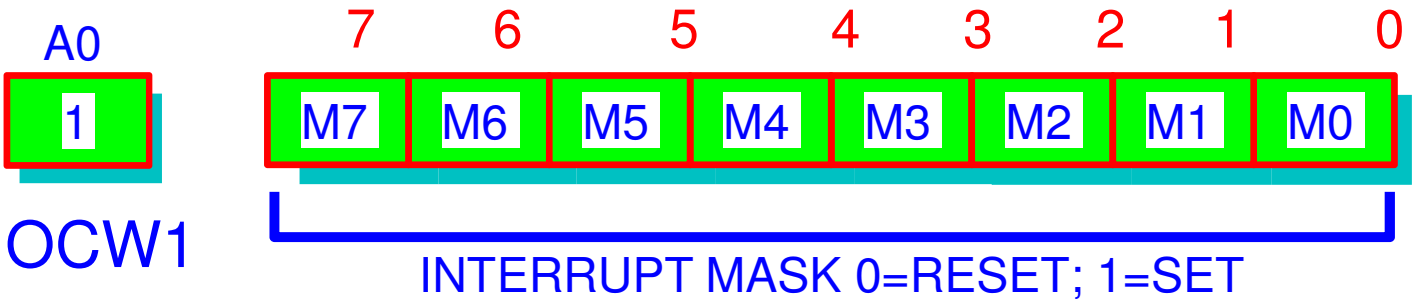


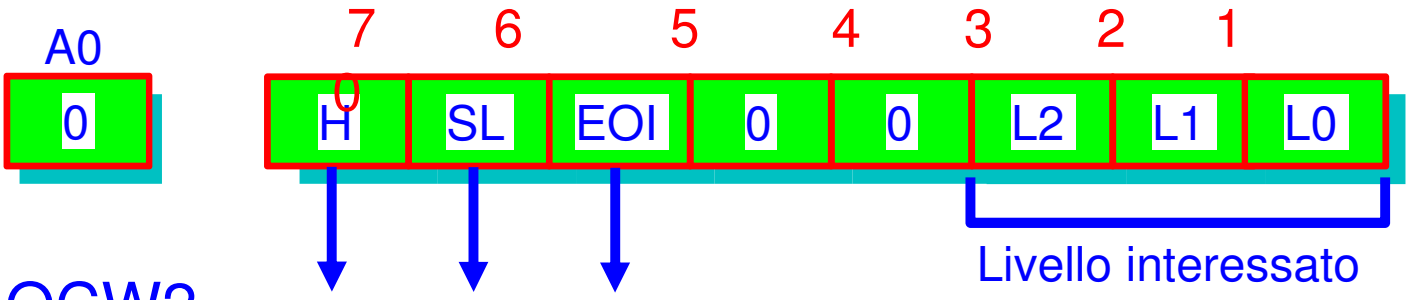
1=AEOI
0=Norm.EOI

1=Special Fully Nested Mode
0=Normal Mode

0 X Unbuffered
1 0 Buffered Slave
1 1 Buffered Master

PAROLE DI CONTROLLO 8259

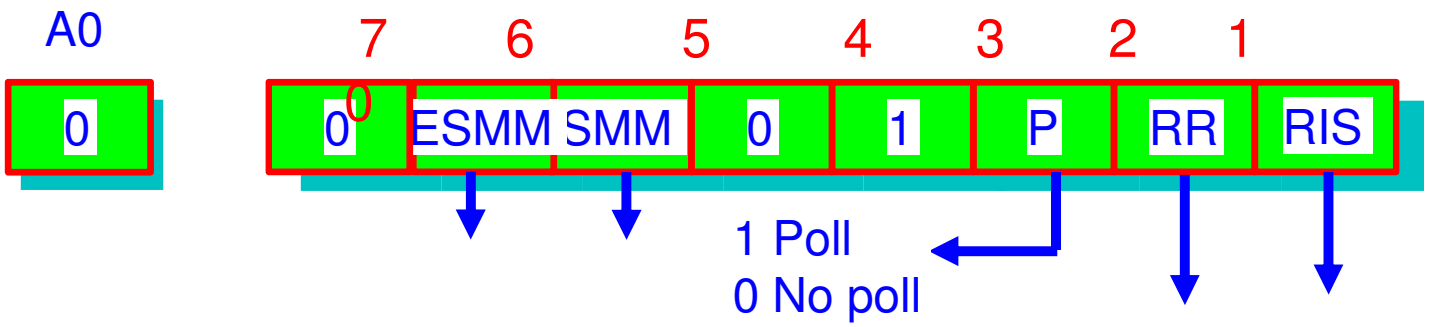




OCW2

0	0	1	EOI non specifico
0	1	1	EOI specifico [liv]
1	0	1	Rotazione **su** EOI non specifico
1	0	0	Rotazione su AEOI (set)
0	0	0	Rotazione su AEOI (reset)
1	1	1	Rotazione **su** EOI specifico [liv]
1	1	0	Set priorità (minima) [liv]
0	1	0	NOP

OCW3



NOP	0	0
NOP	0	1
Reset SMM	1	0
Set SMM	1	1

NOP	0	0
NOP	0	1
Read IRR on RD*	1	0
Read ISR on RD*	1	1

ESMM → Enable Special Mask Mode

SMM → Special Mask Mode: sente tutte le richieste IRi

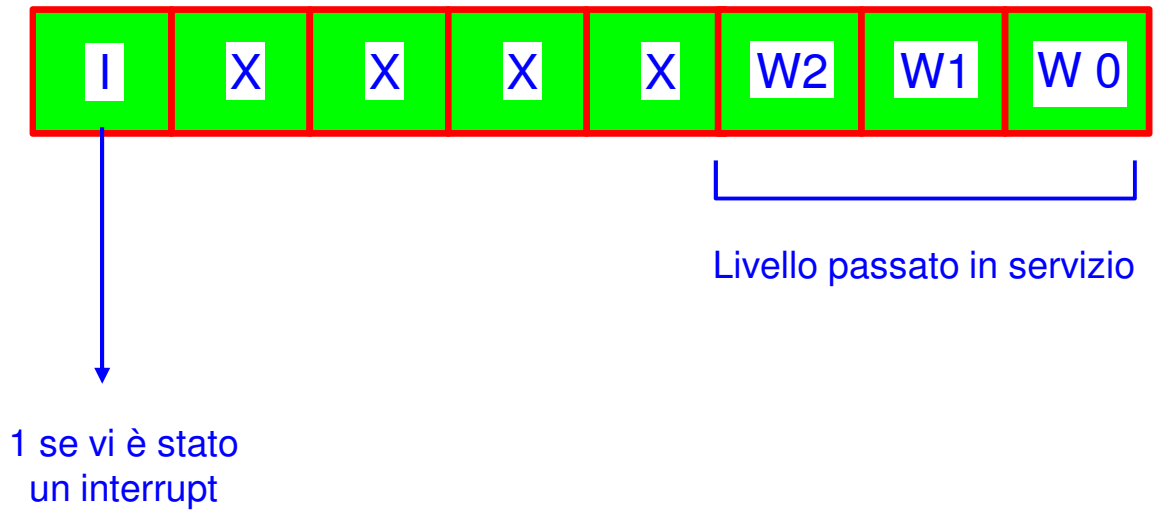
PAROLE DI CONTROLLO 8259

- La modifica della priorità avviene solo mediante ROTAZIONE
- Se AEOI, non appena ISR va a 1 (al II inta) IRR viene resettato: più semplice, ma va usato con cautela
- Se la struttura di priorità dell'interrupt controller è mantenuta anche nel software allora EOI non specifico
- In caso contrario EOI specifico indica via software a quale interrupt deve essere mandato l'end

8259 POLL

- Il comando di POLL ha il seguente effetto: alla prima lettura dell'8259 (CS*=RD*=0) si attiva un meccanismo interno identico che provoca il passaggio in servizio dell'IR di massima priorità (se ce n'è uno pendente). In tal caso il bit 7 del dato letto ha il valore 1, e nei tre bit meno significativi è presente la codifica binaria del livello di interruzione

Parola letta dopo un POLL



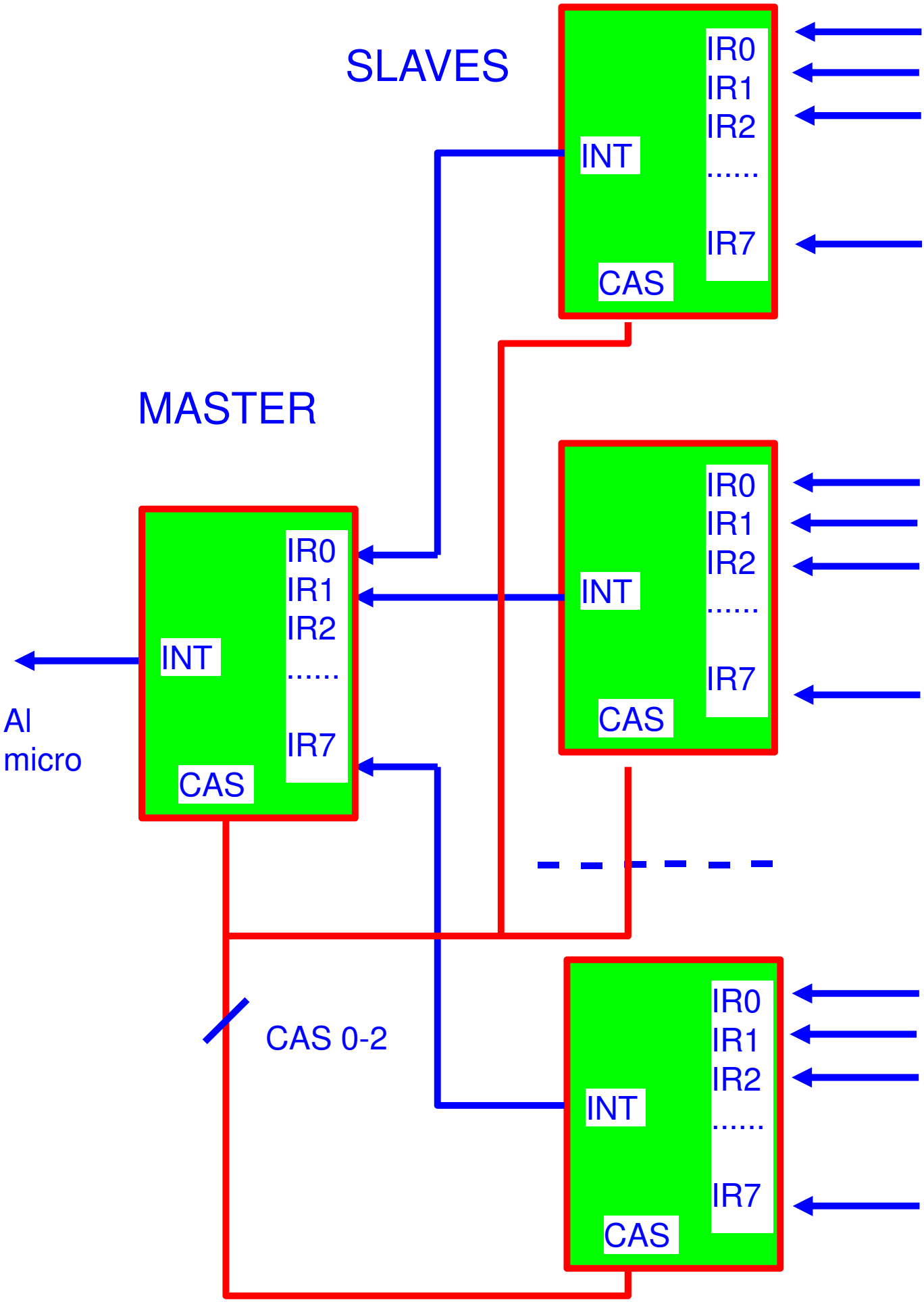
8259 MASTER e SLAVE

configurazione master e slave
ICW1 segnala l'esistenza di una cascata di 8259
ICW3 indica che al segnale IRx e' collegato uno slave

master/slave:

in hardware: (D3,D4)=(0,0) in ICW4, allora SP'/EN come input: 0 (massa), slave; 1(Vcc), master

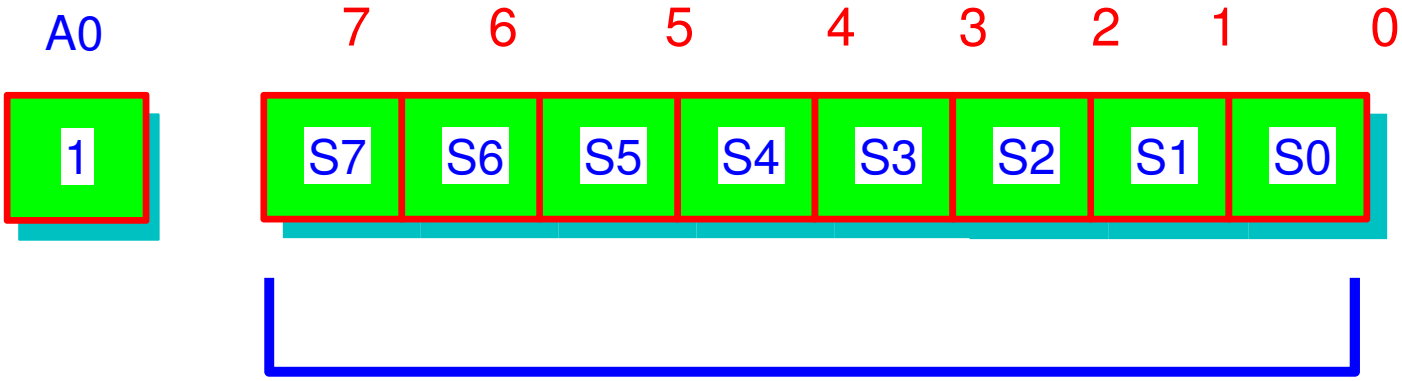
in software (D3,D4)=(11), master; (10), slave e SP'/EN in output indica le temporizzazioni del vettore che l'8259 sta emettendo; serve per pilotare eventuali buffer se 8259 in buffered mode


SLAVES
MASTER
IR0
IR1
IR2
.....
IR7
INT
CAS
IR0
IR1
IR2
.....
IR7
INT
CAS
IR0
IR1
IR2
.....
IR7
INT
CAS
Al
micro
CAS 0-2

8259 IN CASCATA

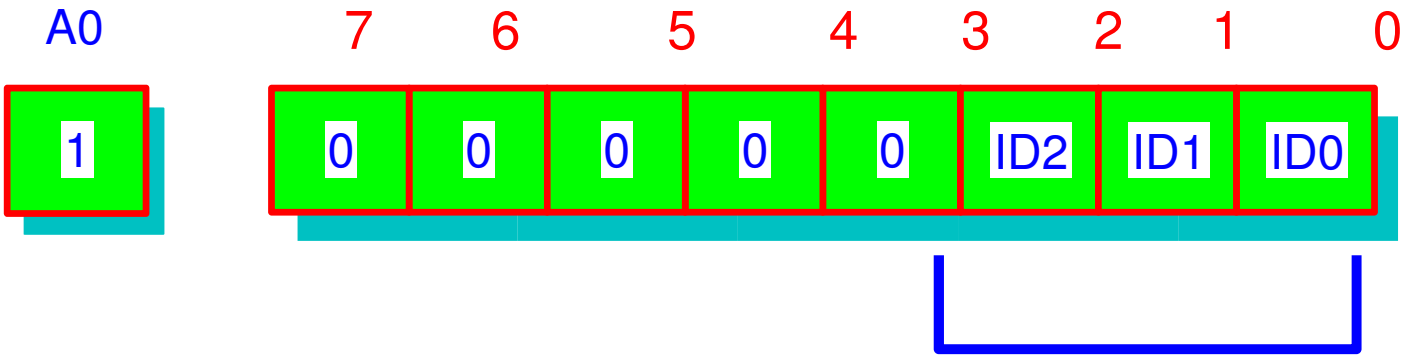
1. INTA* va a tutti i dispositivi
2. Nella struttura in cascata emette l'interrupt TYPE lo slave selezionato con i CASi
3. Il MASTER emette l'interrupt TYPE solo per livelli NON collegati a SLAVES (con i CAS tutti a zero)
4. I CAS sono emessi dal MASTER e ricevuti dagli SLAVES
5. Il livello 0 del MASTER può essere utilizzato solo con 64 interrupts (per il punto 3)
6. La funzione di MASTER o SLAVE può essere definita via hardware (SP/EN*) o via software (ICW4)
7. La struttura in cascata RICHIEDE la ICW3
8. Il meccanismo Special Fully Nested Mode (ICW4 MASTER) mantiene la struttura di priorità completa anche nella cascata
9. Ogni dispositivo (MASTER o SLAVE) deve essere programmato individualmente

ICW3 MASTER



Si = 0 Assenza SLAVE
Si = 1 Presenza SLAVE

ICW3 SLAVE



CASi

DRIVER 8259

```
;----------------8259----------------------
;indirizzi e parole di controllo dell'8259 master
pic_addr       equ       ____h;indirizzo 8259

icw1           equ    0001_0_1b;icw1 di indirizzo add+o
icw2           equ    _____000b;icw2 di indirizzo add+2
icw3           equ    _________b;icw3 di indirizzo add+2
                                        ;NO SE SOLO MASTER
icw4           equ    000_.._1b;icw4 di indirizzo add+2

ocw1           equ    _________b;ocw1 di indirizzo add+2
ocw2           equ    ___00___b;ocw2 di indirizzo add+0
ocw3           equ    0..01...b;ocw3 di indirizzo add+0
;
;indirizzi e parole di controllo dell'8259 slave
pics_addr      equ       ____h;indirizzo 8259 slave

icws1          equ    0001_011b;icw1 di indirizzo add+0
icws2          equ    _____000b;icw2 di indirizzo add+2
icws3          equ    00000___b;icw3 di indirizzo add+2
icws4          equ    000....1b;icw4 di indirizzo add+2

ocws1          equ    _________b;ocw1 di indirizzo add+2
ocws2          equ    ___00___b;ocw2 di indirizzo add+0
ocws3          equ    0__01...b;ocw3 di indirizzo add+0
```

Driver 82C59 in C

```
//File Header
//MASTER
#define 82C59Addr  0xYYYY; //indirizzo 8259

#define ICW1  0001Y0Y1b; //indirizzo ICW1 = 82C59Addr+0
#define ICW2  YYYYY000b; //indirizzo ICW2 = 82C59Addr+2
#define ICW3  YYYYYYYYb; //indirizzo ICW3 = 82C59Addr+2
#define ICW4 000YYYY1b; //indirizzo ICW4 = 82C59Addr+2

#define OCW1  YYYYYYYYb; //indirizzo OCW1 = 82C59Addr+2
#define OCW2  YYY00YYYb; //indirizzo OCW2 = 82C59Addr+0
#define OCW3  0YY01YYYb; //indirizzo OCW3 = 82C59Addr+0

//SLAVE
#define 82C59SAddr  0xYYYY; //indirizzo 8259

#define ICW1S  0001Y011b; //indirizzo ICW1 = 82C59SAddr+0
#define ICW2S  YYYYY000b; //indirizzo ICW2 = 82C59SAddr+2
#define ICW3S  00000YYYb; //indirizzo ICW3 = 82C59SAddr+2
#define ICW4S 000YYYY1b; //indirizzo ICW4 = 82C59SAddr+2

#define OCW1S  YYYYYYYYb; //indirizzo OCW1=82C59SAddr+2
#define OCW2S  YYY00YYYb; //indirizzo OCW2 = 82C59SAddr+0
#define OCW3S  0YY01YYYb; //indirizzo OCW3 = 82C59SAddr+0
```

DRIVER 8259

```
;inizializzo INTERRUPT CONTROLLER 8259
INIZ_59 proc near
        mov DX,PIC_ADDR
        mov AL,ICW1
        out DX,AL
        mov DX,PIC_ADDR+2
        mov AL,ICW2
        out DX,AL
        mov AL,ICW3
        out DX,AL
        mov AL,ICW4
        out DX,AL
        mov AL,OCW1
        out DX,AL
        mov DX,PIC_ADDR
        mov AL,OCW2
        out DX,AL
        mov AL,OCW3
        out DX,AL
        ret
INIZ_59 endp
```

```
;inizializzo  8259 slave
INIZ_59S proc near
        mov DX,PICS_ADDR
        mov AL,ICWS1
        out DX,AL
        mov DX,PICS_ADDR+2
        mov AL,ICWS2
        out DX,AL
        mov AL,ICWS3
        out DX,AL
        mov AL,ICWS4
        out DX,AL
        mov AL,OCWS1
        out DX,AL
        mov DX,PICS_ADDR
        mov AL,OCWS2
        out DX,AL
        mov AL,OCWS3
        out DX,AL
        ret
INIZ_59S endp
```

Driver in C

```
Void Prog82C59(void) //programmazione del MASTER
{
      unsigned int* ExtMemPointer;
      ExtMemPointer = (unsigned int *)82C59Addr; //punto al
                                                  //registro di comando
      ExtMemPointer[0] = ICW1;
      ExtMemPointer[2] = ICW2; //scrivo la word
      ExtMemPointer[2] = ICW3; //scrivo la word
      ExtMemPointer[2] = ICW4; //scrivo la word
      ExtMemPointer[2] = OCW1; //scrivo la word
      ExtMemPointer[0] = OCW2; //scrivo la word
      ExtMemPointer[0] = OCW3; //scrivo la word
}

Void Prog82C59S(void) //programmazione del MASTER
{
      unsigned int* ExtMemPointer;
      ExtMemPointer = (unsigned int *)82C59SAddr; //punto al
                                                  //registro di comando
      ExtMemPointer[0] = ICW1S;
      ExtMemPointer[2] = ICW2S; //scrivo la word
      ExtMemPointer[2] = ICW3S; //scrivo la word
      ExtMemPointer[2] = ICW4S; //scrivo la word
      ExtMemPointer[2] = OCW1S; //scrivo la word
      ExtMemPointer[0] = OCW2S; //scrivo la word
      ExtMemPointer[0] = OCW3S; //scrivo la word
}
```