Calcolatori Elettronici T
Ingegneria Informatica

06 Programmable Interrupt Controller (PIC)

Stefano Mattoccia

Gestione delle interruzioni con PIC

- Abbiamo già visto che è possibile, opzionalmente, utilizzare un dispositivo *ad-hoc* per la gestione di multiple sorgenti interruzioni denominato PIC

- Il PIC velocizza e facilita la fase di analisi e gestione degli interrupt

- Tipicamente un PIC consente di:
  - Abilitare disabilitare singole interruzioni
  - Fornire informazioni sulle interruzioni asserite
  - Gestire la priorità delle interruzioni

- Per le ragioni evidenziate, un PIC è programmabile mediante l'utilizzo di opportuni registri interni

- Sebbene sia sempre possibile una gestione interamente software delle interruzioni, l'utilizzo di un PIC può essere una valida alternativa

- Per queste ragioni, progettiamo un PIC molto semplice

Progetto di un semplice PIC

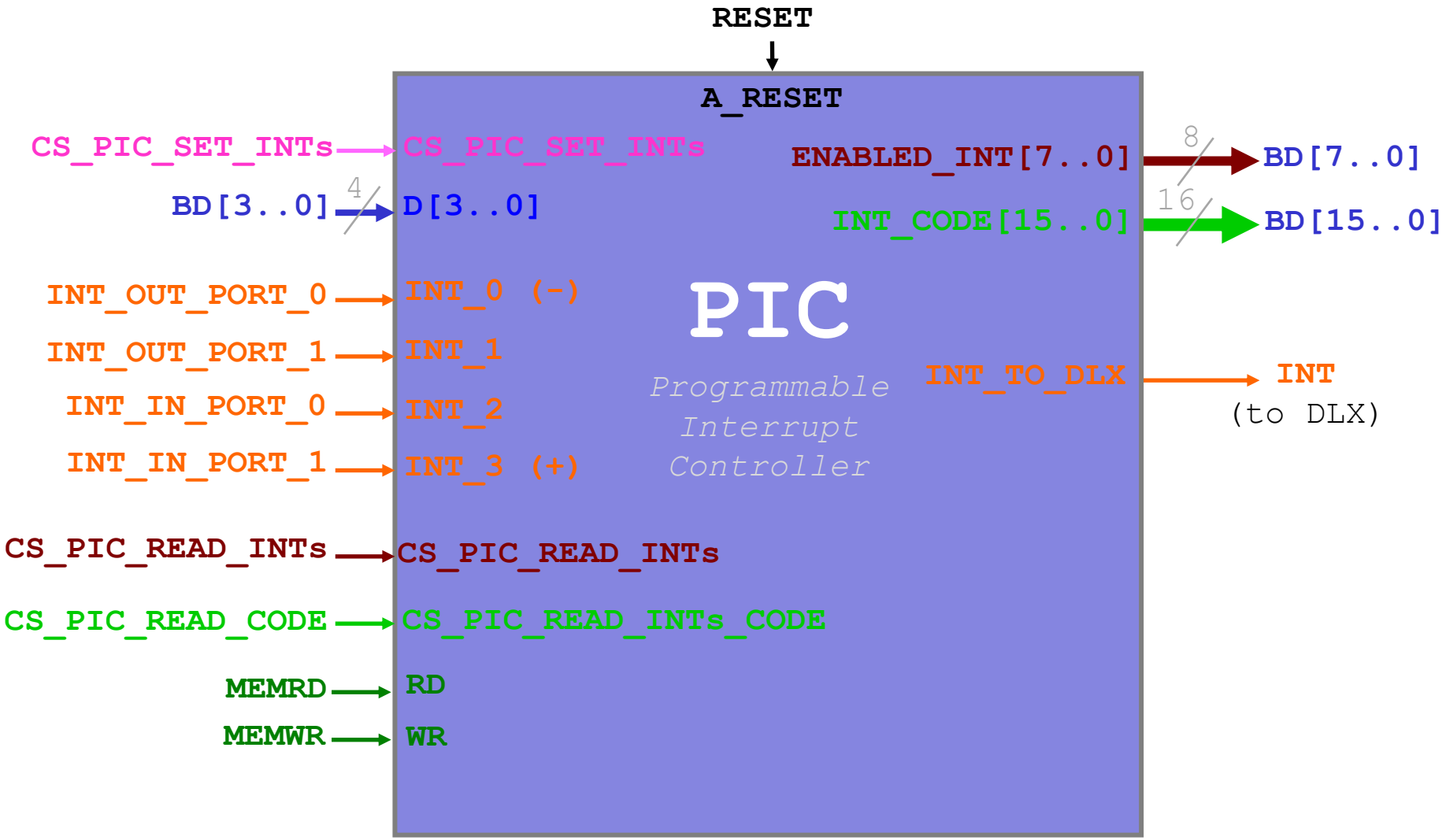
In un sistema basato sul **DLX**, con **1 GB di EPROM** mappata negli indirizzi bassi e **512 MB di RAM** mappata negli indirizzi alti, sono presenti e già progettate **2 porte a 8 bit in INPUT (IN_1** e **IN_0)** e **2 porte a 8 bit in OUTPUT (OUT_1** e **OUT_0)** basate sul protocollo di *handshake*. **Progettare un semplice PIC** al fine di assegnare le seguenti priorità statiche alle quattro interruzioni: **IN_1 (massima priorità)**, **IN_0**, **OUT_1** e **OUT_0 (minima priorità)**. Il PIC dovrà inoltre consentire di:

a) **disabilitare/abilitare selettivamente**, mediante parole di controllo, **ciascuna interruzione generata dalle 4 periferiche**
b) **fornire le interruzioni asserite** (tra quelle abilitate)
c) **fornire un codice che indica qual è l'interruzione più prioritaria** (tra quelle abilitate) **in un determinato istante**

Utilizzando la rete logica progettata gestire le quattro interruzioni in modo che durante l'esecuzione dell'*interrupt handler* sia eseguito, nel modo più rapido possibile, **solo il trasferimento attivo più prioritario in quel momento**. Eventuali altre richieste di trasferimento attive saranno gestite durante successive esecuzioni dell'interrupt handler. **I dati letti dalle porte in INPUT** dovranno essere scritti a **FFFF0080 (IN_1)** e **FFFF0040 (IN_0)** mentre **i dati da scrivere nelle porte in OUTPUT** dovranno essere letti da **FFFF0020 (OUT_1)** e **FFFF0010 (OUT_0)**. **All'avvio del sistema il PIC dovrà automaticamente disabilitare tutte le richieste di interruzione** provenienti dalle quattro porte.

- Scrivere **il codice che abilita tutte le interruzioni nel** PIC e **il codice che consente di leggere lo stato degli interrupt**
- Scrivere il **codice ottimizzato dell'*interrupt handler*** (i registri da R25 a R29 possono essere utilizzati senza la necessità di doverli ripristinare).

Il **PIC** (*Programmable Interrupt Controller*), in grado di gestire 4 interruzioni, può essere schematizzato nel modo seguente:



Nel PIC, la massima priorità è assegnata a **INT_3**, quella minima a **INT_0**. Le priorità sono statiche, come previsto dal testo.

I segnali di ingresso del PIC **INT_3**, **INT_2**, **INT_1** e **INT_0** sono connessi ai 4 interrupt delle periferiche in modo da soddisfare i vincoli sulla priorità previsti dal testo del problema.

Scrivendo a **CS_PIC_SET_INTs**, i dati presenti sui pin **D[3..0]** consentono di abilitare/disabilitare i singoli interrupt.

Gli interrupt asseriti dalle periferiche, tra quelli abilitati, possono essere letti, a **CS_PIC_READ_INTs**, attraverso i segnali **ENABLED_INT[7..0]**. Essendo previsti solo 4 interrupt, 4 degli 8 bit sono cablati a 0 (i 4 bit più significativi).

Il codice che corrisponde all'interrupt più prioritario, tra quelli abilitati, potrà essere letto, a **CS_PIC_READ_CODE**, attraverso i segnali **INT_CODE[15..0]**. Di questi ultimi 16 segnali, 12 saranno sempre cablati a 0 per ragioni chiarite in seguito.

Dispositivi e segnali presenti nel sistema

Memorie:

RAM_512_MB mappata da E0000000h:FFFFFFFFh, 4 banchi da 128 MB
EPROM_1_GB mappata da 00000000h:3FFFFFFFh, 4 banchi da 256 MB

Porte di input, output e altri chip-select e/o segnali:

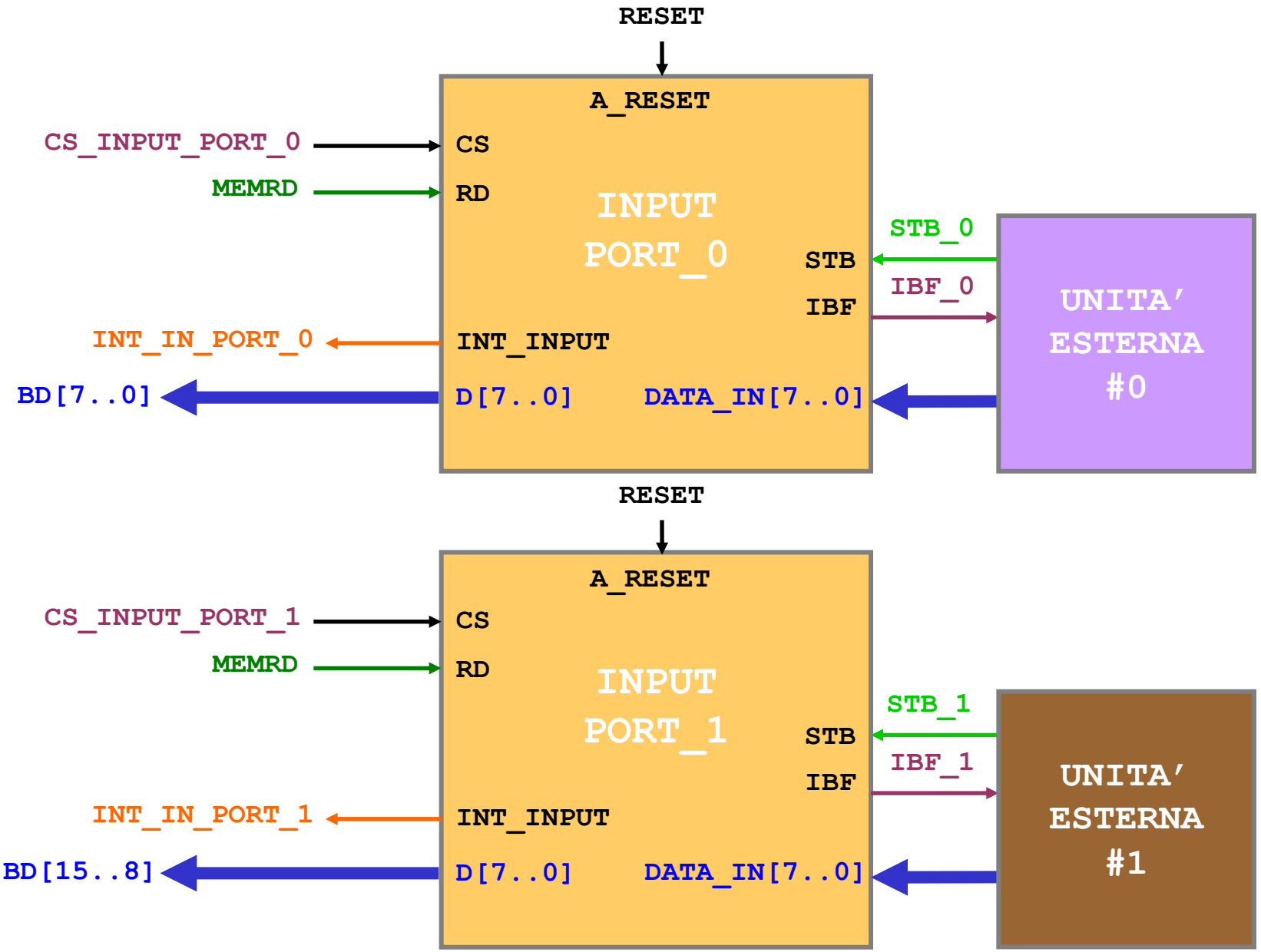
CS_INPUT_PORT_0 mappato a 80000000h
CS_INPUT_PORT_1 mappato a 80000001h
CS_OUTPUT_PORT_0 mappato a 80000002h
CS_OUTPUT_PORT_1 mappato a 80000003h

CS_PIC_SET_INTs mappato a 80000004h
CS_PIC_READ_INTs mappato a 80000008h
CS_PIC_READ_CODE mappato a 8000000Ch

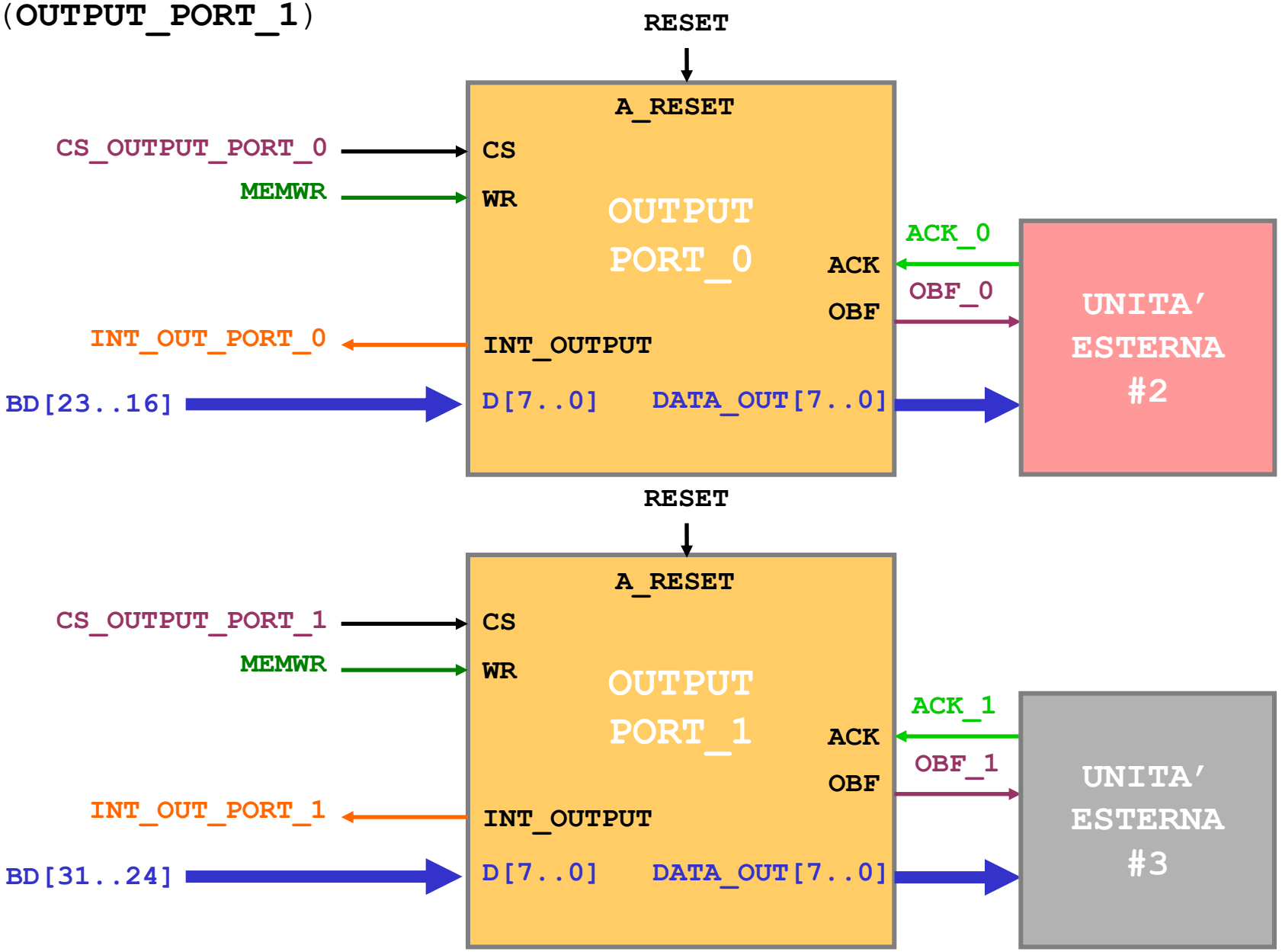
Segnali di decodifica di memorie, periferiche e segnali:

```
CS_RAM_0           = BA31·BA30·BE0
CS_RAM_1           = BA31·BA30·BE1
CS_RAM_2           = BA31·BA30·BE2
CS_RAM_3           = BA31·BA30·BE3

CS_INPUT_PORT_0    = BA31·BA30*·BA3*·BA2*·BE0·IBF_0   mappato a 80000000h
CS_INPUT_PORT_1    = BA31·BA30*·BA3*·BA2*·BE1·IBF_1   mappato a 80000001h
CS_OUTPUT_PORT_0   = BA31·BA30*·BA3*·BA2*·BE2·OBF_0*  mappato a 80000002h
CS_OUTPUT_PORT_1   = BA31·BA30*·BA3*·BA2*·BE3·OBF_1*  mappato a 80000003h

CS_PIC_SET_INTs    = BA31·BA30*·BA3*·BA2              mappato a 80000004h
CS_PIC_READ_INTs   = BA31·BA30*·BA3·BA2*·MEMRD        mappato a 80000008h
CS_PIC_READ_CODE   = BA31·BA30*·BA3·BA2·MEMRD         mappato a 8000000Ch

CS_EPROM_0         = BA31*·BE0
CS_EPROM_1         = BA31*·BE1
CS_EPROM_2         = BA31*·BE2
CS_EPROM_3         = BA31*·BE3
```

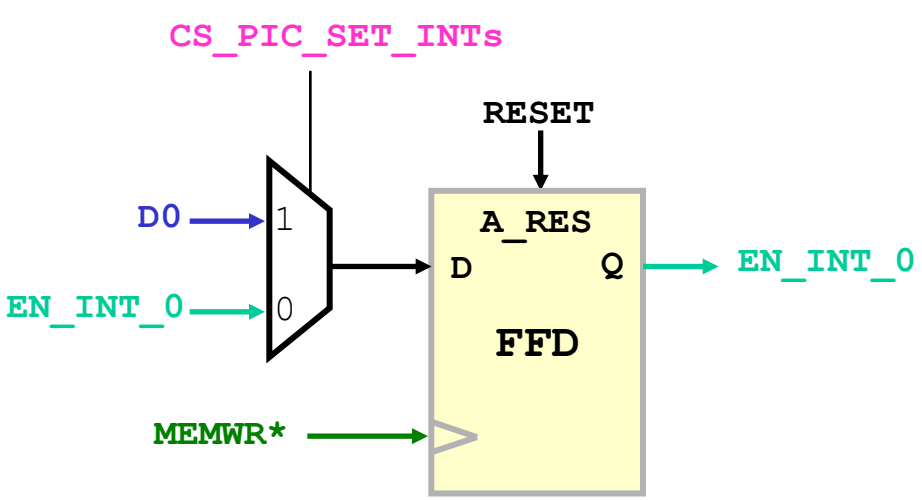
Nel sistema sono presenti due porte in input, collegate ai bus dati **BD[7..0]** (**INPUT_PORT_0**) e **BD[15..8]** (**INPUT_PORT_1**)

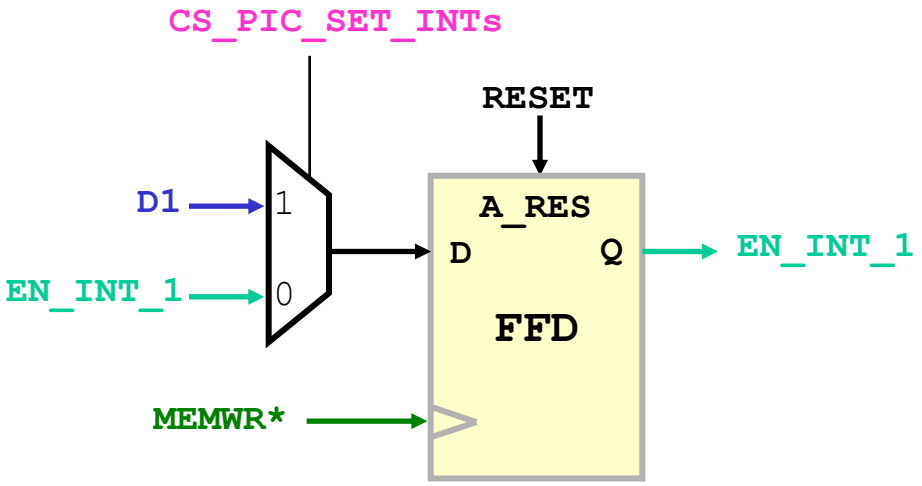


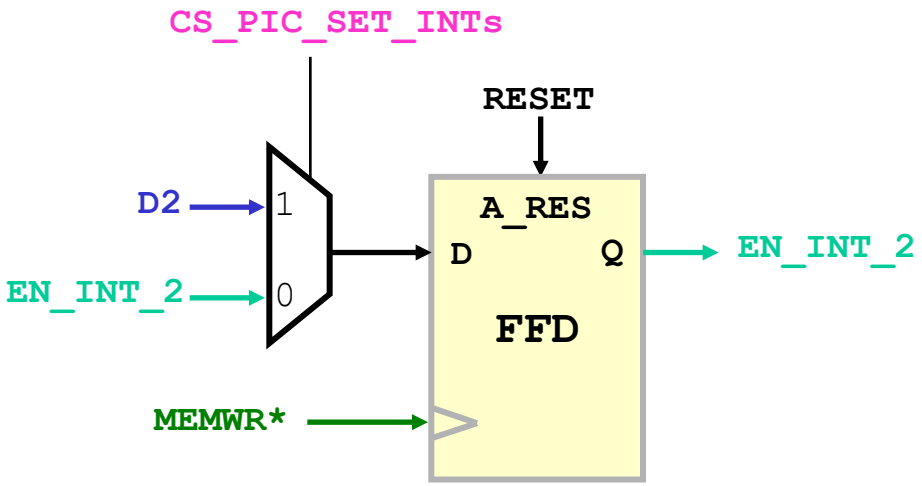
Nel sistema sono presenti anche due porte in output collegate ai bus dati **BD[23..16]** (**OUTPUT_PORT_0**) e **BD[31..24]** (**OUTPUT_PORT_1**)

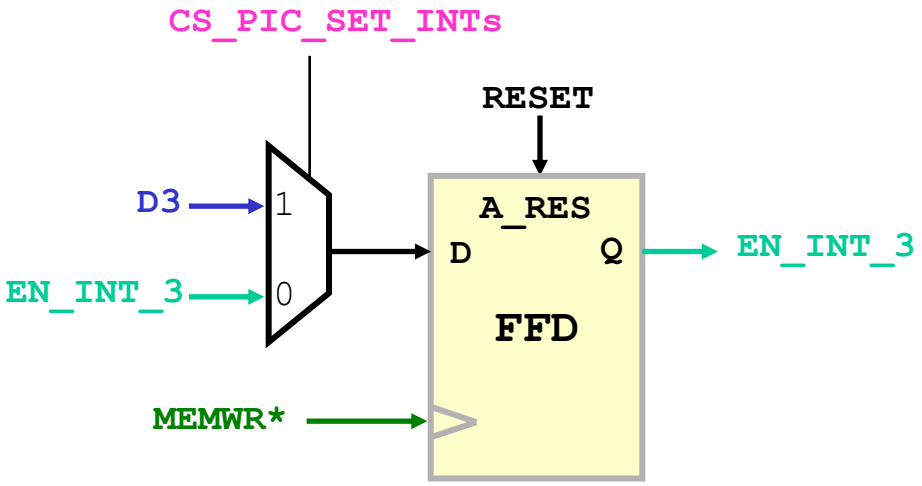


Al fine di **abilitare** e **disabilitare selettivamente gli interrupt** si possono utilizzare quattro FFD. I quattro bit **D[3..0]** sono connessi ai segnali **BD[3..0]** del bus dati e utilizzati per condizionare ogni singola interruzione mediante i segnali **EN_INT_0**,**EN_INT_1**,**EN_INT_2** e **EN_INT_3** generati dalle reti seguenti:

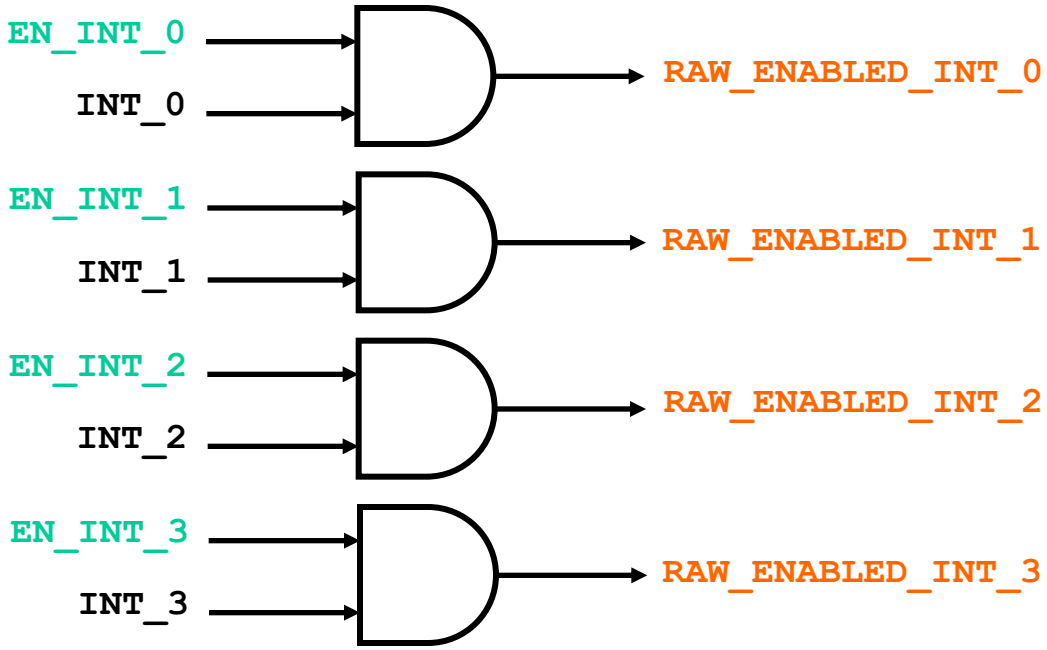








Di conseguenza, in ogni istante, gli interrupt abilitati risultano dalle uscite di questa rete:



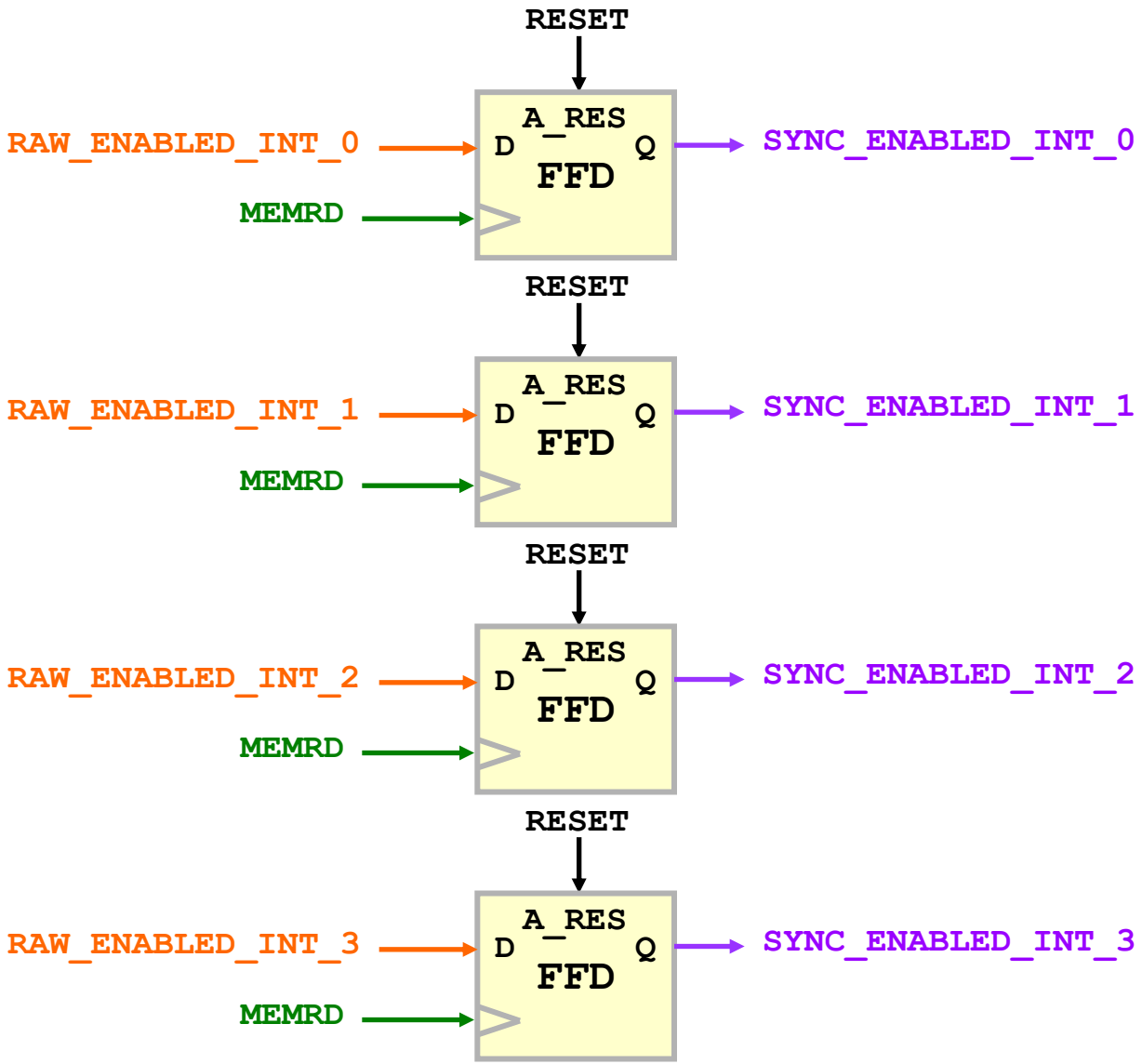
Si ricorda che, come mostrato nello schema ai morsetti del PIC, risulta:

INT_0 = INT_OUT_PORT_0
INT_1 = INT_OUT_PORT_1
INT_2 = INT_IN_PORT_0
INT_3 = INT_IN_PORT_1

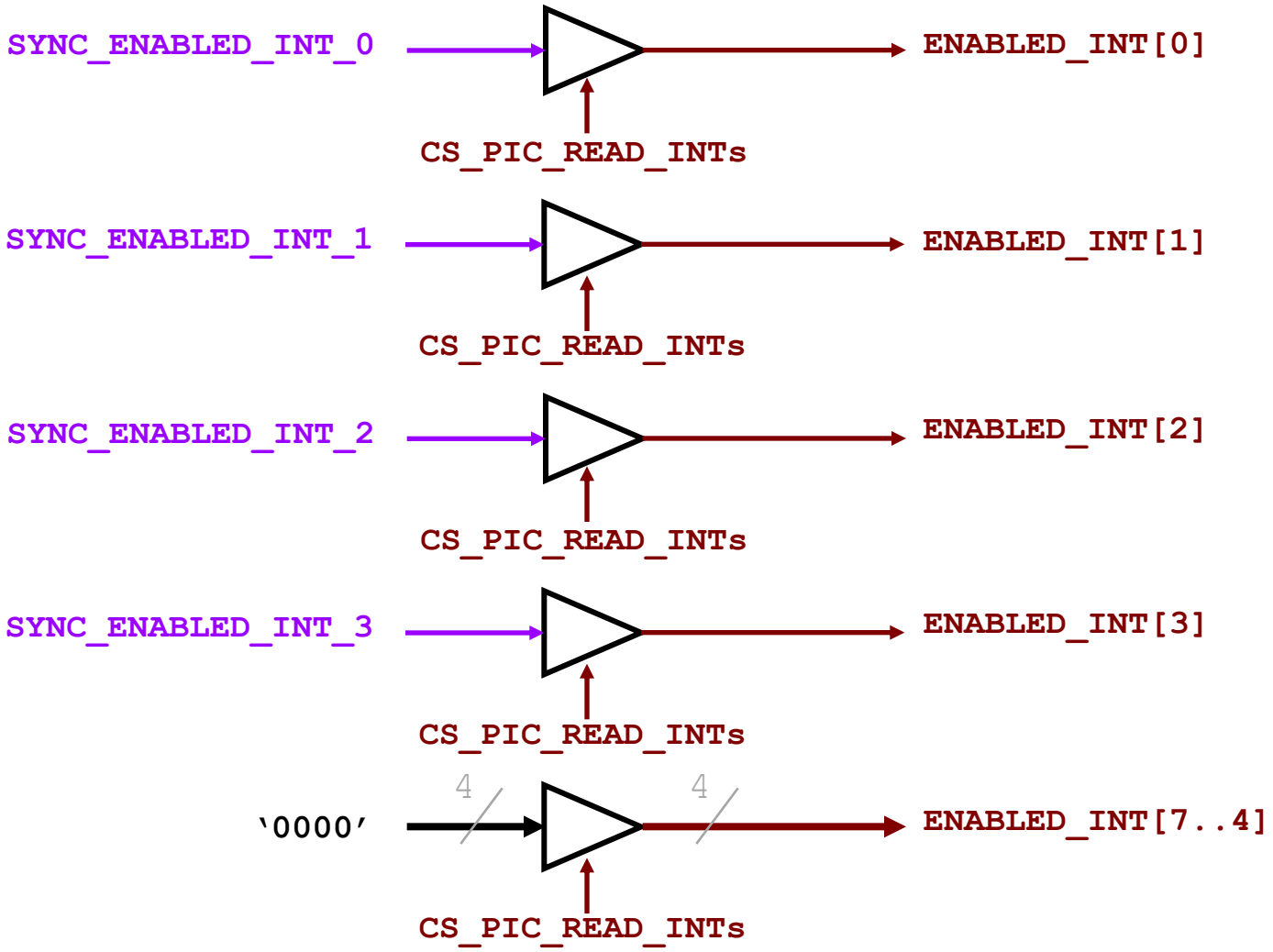
Al pin di interrupt del DLX è pertanto inviato il segnale:

INT_DLX = RAW_ENABLED_INT_0 + RAW_ENABLED_INT_1 +
RAW_ENABLED_INT_2 + RAW_ENABLED_INT_3

Per evitare problemi di *metastabiità* è possibile campionare lo stato degli interrupt, prima di procedere alla loro lettura, per esempio con quattro FFD che campionano gli interrupt sul fronte di salita di **MEMRD**.

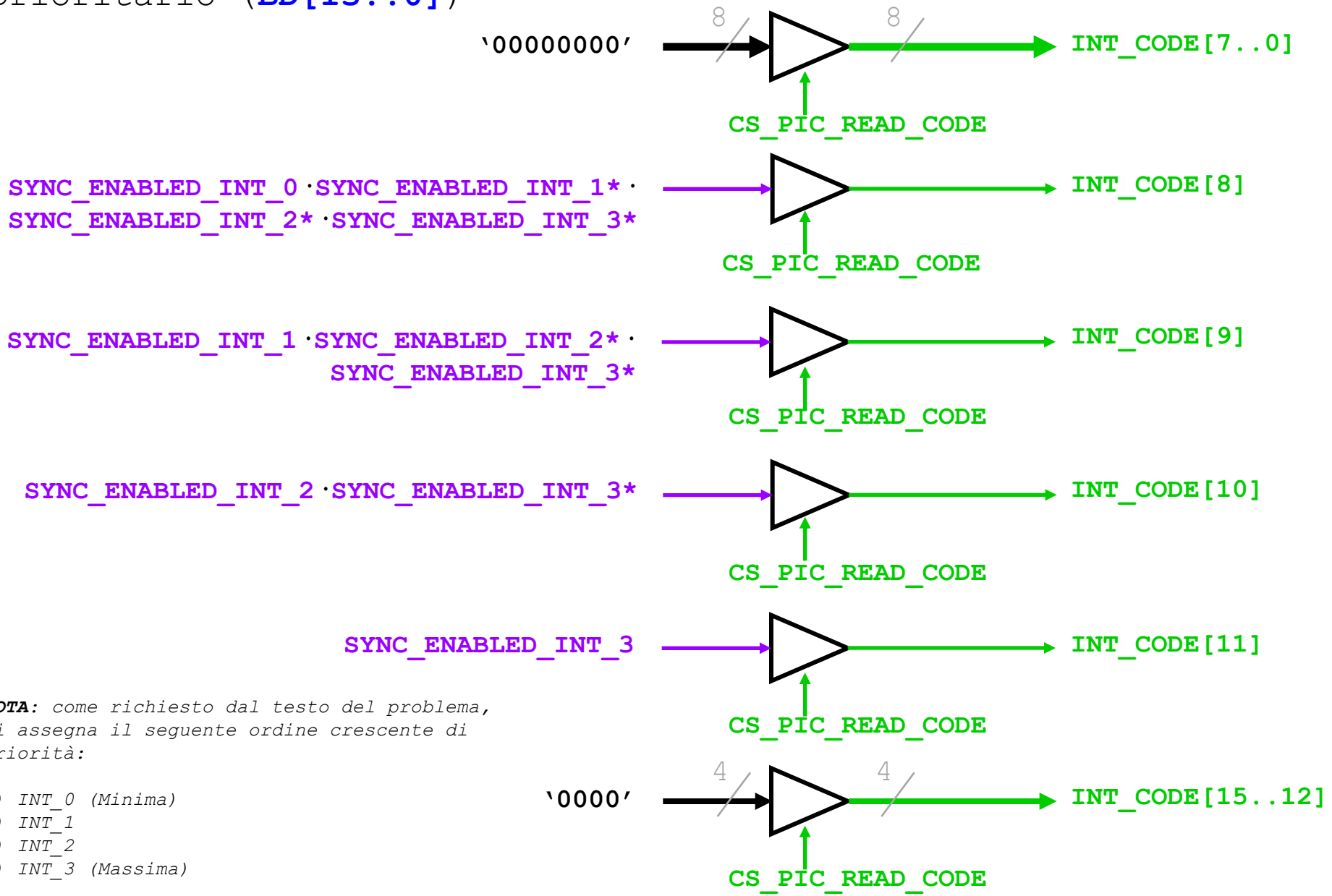


Al fine di leggere lo stato degli interrupt asseriti, tra quelli che sono stati abilitati, si utilizzano dei buffer *tri-state* pilotati dal segnale **CS_PIC_READ_INTs** come segue:



I segnali **ENABLED_INT[7..0]** sono connessi al bus dati **BD[7..0]**

La rete seguente, consente di leggere a **CS_PIC_READ_CODE** il codice a **16 bit** (per ragioni mostrate dopo) dell'interrupt più prioritario (**BD[15..0]**)



NOTA: *come richiesto dal testo del problema, si assegna il seguente ordine crescente di priorità:*

0) INT_0 (Minima)
1) INT_1
2) INT_2
3) INT_3 (Massima)

I **codici di priorità**, a **16 bit** per velocizzare l'handler, associati alle **quattro interruzioni** e letti all'indirizzo **CS_PIC_READ_CODE**, risultano:

0800h se è asserito **SYNC_ENABLED_INT_3** (massima priorità)

0400h se è asserito **SYNC_ENABLED_INT_2** e non **SYNC_ENABLED_INT_3**

0200h se è asserito **SYNC_ENABLED_INT_1** e non **SYNC_ENABLED_INT_2** o **SYNC_ENABLED_INT_3**

0100h se è asserito **SYNC_ENABLED_INT_0** e nessun altro segnale

Il codice per abilitare le interruzioni dalle 4 porte risulta:

```
LHI R25,8000h          ; R25 = 80000000h
ADDI R26,R0,000Fh      ; R26 = 0 + 0000000F
SB  R26,(R25)04h       ; scrive il byte 0Fh contenuto in R26
                       ; all'indirizzo CS_PIC_SET_INTs (80000004h)
```

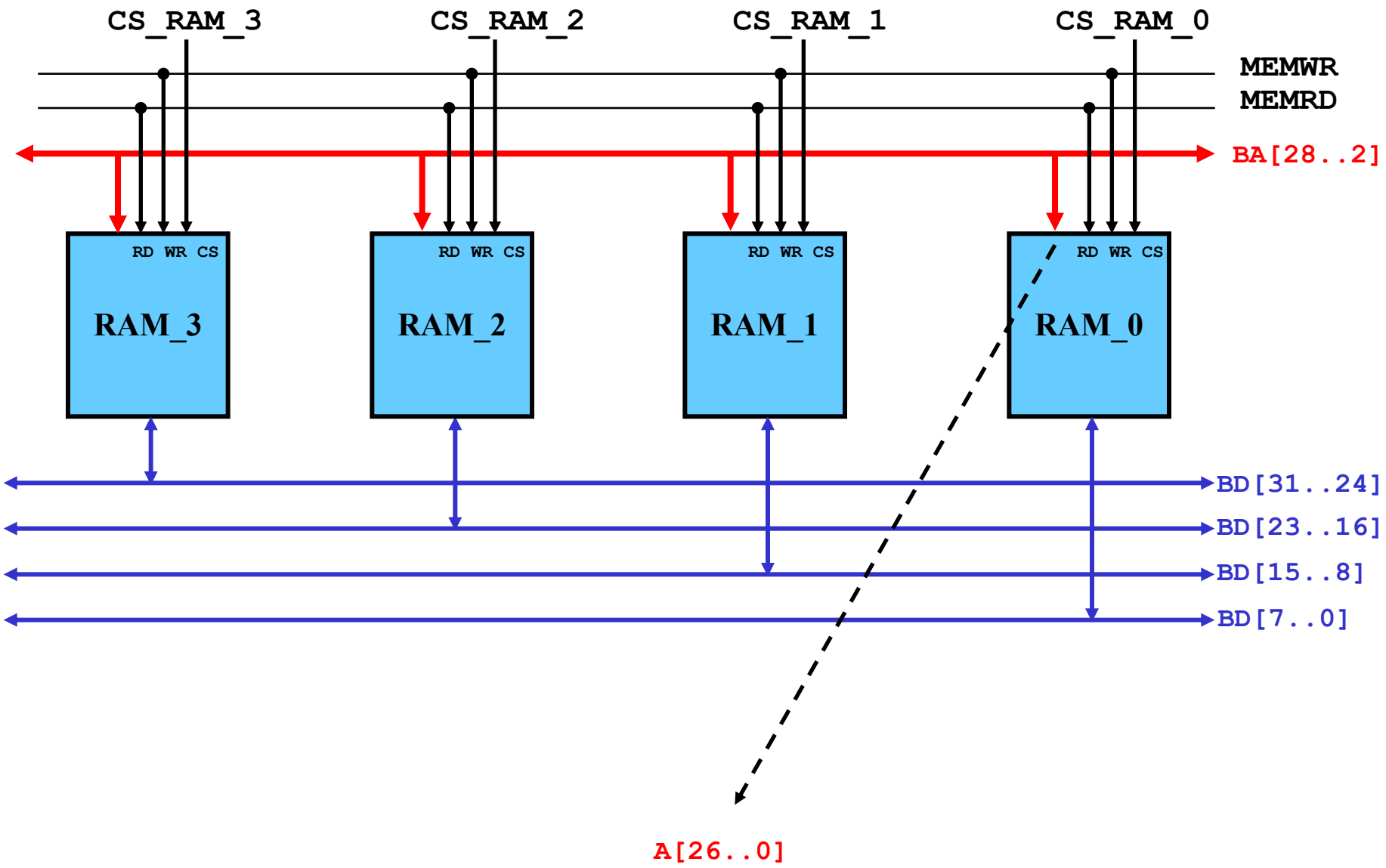
Il codice per leggere quali sono le interruzioni asserite:

```
LHI R25,8000h          ; R25 = 80000000h
LBU R26,(R25)08h       ; legge in R26 gli interrupt asseriti, tra quelli
                       ; abilitati, all'indirizzo CS_PIC_READ_INTs
                       ; (80000008h)
```

Il codice dell'*interrupt handler* è il seguente:

```
00000000: LHI R25,8000h        ; prepara indirizzo 80000000h
00000004: LHU R26,(R25)0Ch     ; lettura del codice di priorità a 16 bit
                               ; all'indirizzo CS_PIC_READ_CODE
00000008: LHI  R27,FFFF        ; prepara in R27 l'indirizzo per
                               : operazioni comuni successive al salto
0000000C: JR   R26             ; salta all'indirizzo presente in R26
                               ; che corrisponde al codice di interrupt
                               ; più prioritario letto mediante LHU

00000100: LBU R28,(R27)10h     ; legge in memoria un byte a FFFF0010h
00000104: SB R28,(R25)2h       ; scrive quanto letto in OUTPUT_PORT_0
00000108: RFE                  ; (80000002h) e ritorna dall'interrupt

00000200: LBU R28,(R27)20h     ; legge in memoria un byte a FFFF0020h
00000204: SB R28,(R25)3h       ; scrive quanto letto in OUTPUT_PORT_1
00000208: RFE                  ; (80000003h) e ritorna dall'interrupt

00000400: LBU R28,(R25)0       ; legge da INPUT_PORT_0 (80000000h)
00000404: SB R28,(R27)40h      ; scrive byte in memoria a FFFF0040h
00000408: RFE                  ; ritorna dall'interrupt

00000800: LBU R28,(R25)1       ; legge da INPUT_PORT_1 (80000001h)
00000804: SB R28,(R27)80h      ; scrive byte in memoria a FFFF0080h
00000808: RFE                  ; ritorna dall'interrupt
```

Interfacciamento RAM



Interfacciamento EPROM