Versione N° 1.2

LaurTec

ATmega 2560

Corso di programmazione per microcontrollori Atmel

Autore : *Marcello Pinna*

ID: UT0005-IT

Copyright © 2020 Mauro Laurenti

INFORMATIVA

Come prescritto dall'art. 1, comma 1, della legge 21 maggio 2004 n.128, l'autore avvisa di aver assolto, per la seguente opera dell'ingegno, a tutti gli obblighi della legge 22 Aprile del 1941 n. 633, sulla tutela del diritto d'autore.
Tutti i diritti di questa opera sono riservati. Ogni riproduzione ed ogni altra forma di diffusione al pubblico dell'opera, o parte di essa, senza un'autorizzazione scritta dell'autore, rappresenta una violazione della legge che tutela il diritto d'autore, in particolare non ne è consentito un utilizzo per trarne profitto.
La mancata osservanza della legge 22 Aprile del 1941 n. 633 è perseguibile con la reclusione o sanzione pecuniaria, come descritto al Titolo III, Capo III, Sezione II.
A norma dell'art. 70 è comunque consentito, per scopi di critica o discussione, il riassunto e la citazione, accompagnati dalla menzione del titolo dell'opera e dal nome dell'autore.

AVVERTENZE

I progetti presentati non hanno la marcatura CE, quindi non possono essere utilizzati per scopi commerciali nella Comunità Economica Europea.

Chiunque decida di far uso delle nozioni riportate nella seguente opera o decida di realizzare i circuiti proposti, è tenuto pertanto a prestare la massima attenzione in osservanza alle normative in vigore sulla sicurezza.
L'autore declina ogni responsabilità per eventuali danni causati a persone, animali o cose derivante dall'utilizzo diretto o indiretto del materiale, dei dispositivi o del software presentati nella seguente opera.
Si fa inoltre presente che quanto riportato viene fornito così com'è, a solo scopo didattico e formativo, senza garanzia alcuna della sua correttezza.

L'autore ringrazia anticipatamente per la segnalazione di ogni errore.

Tutti i marchi citati in quest'opera sono dei rispettivi proprietari.

Indice

Capitolo VI....4
Le Interruzioni....4
Eventi d'interruzione esterni....8
EICRA: External Interrupt Control Register A - INT0-INT3....8
EICRB: External Interrupt Control Register B - INT4-INT7....8
EIMSK: External Interrupt Mask Register....9
EIFR: External Interrupt Flag Register....9
PCICR: Pin Change Interrupt Control Register....10
PCIFR: Pin Change Interrupt Flag Register....10
PCMSK0: Pin Change Mask Register 0....11
PCMSK1: Pin Change Mask Register 1....11
PCMSK2: Pin Change Mask Register 2....11
Esempi d'impostazione di un interrupt esterno....12
History....15

Capitolo VI

Le Interruzioni

Un interrupt è un evento che interrompe momentaneamente la CPU dall'esecuzione di un'operazione, ne fa eseguire altre, e poi la CPU riprende dal punto in cui era stata interrotta. In un certo qual modo un'interruzione potrebbe essere semplicemente una telefonata mentre si sta guardando la televisione. L'interruzione ci porta a rispondere al telefono per poi tornare a guardare la televisione. Come vedremo a breve ci sono anche modi per ignorare le interruzioni a seconda delle impostazioni, per cui se ci è stato detto di non rispondere al telefono, possiamo continuare a guardare la televisione.
Una interruzione può essere generata da eventi esterni al microcontrollore, per esempio per mezzo di hardware come un pulsante, oppure da eventi interni generati dalle periferiche del microcontrollore. Non è detto che un evento d'interruzione debba avvenire necessariamente in maniera sequenziale e in maniera prevedibile. Infatti, possono verificarsi più eventi d'interrupt nello stesso istante o periodo di tempo in cui si stanno eseguendo delle istruzioni relative ad un altro interrupt. Per esempio mentre ci siamo alzati per rispondere al telefono potrebbe suonare il campanello della porta. A questo punto dobbiamo prendere una decisione sul cosa fare.
Per stabilire quale interrupt deve essere eseguito per primo è necessario definire un livello di priorità. In parole povere, un interrupt deve sempre poter prevalere su un altro interrupt. Per esempio, si supponga di avere a bordo di una scheda un microcontrollore che sta eseguendo in quel momento una routine di interrupt, che accende e spegne un LED ogni secondo. Premendo il pulsante di *Reset* viene generato un interrupt esterno a livello hardware che azzera tutto il sistema. Risulta quindi evidente che l'interrupt del *Reset* prevale sull'esecuzione di qualunque software e cioè che il *Reset* ha una priorità maggiore. Nel caso dell'Atmega 2560 questo livello di priorità è stabilito in una tabella chiamata *Reset e Interrupt Vector*, in cui sono elencate tutte le 57 sorgenti di interrupt di cui dispone. I livelli di priorità sono definiti in ordine crescente, ovvero per ogni sorgente i livelli precedenti comandano sempre su quelli successivi. Questa tabella può essere anche alterata nel caso si usi un bootloader, agendo sul registro MCUCR (*MCU Control Register*) e sui *fuses* di configurazione. In questo capitolo questa descrizione non verrà trattata. In Tabella 1 e Tabella 2 sono riportati gli interrupt supportati, con il relativo ordine di priorità. Un *Vector Number* più basso definisce il livello di priorità più alto, per cui al livello 1 troviamo il *Reset*.

Il *Reset* è un evento di interruzione particolare, infatti non può essere ignorato. Questo tipo di interruzioni sono dette non mascherabili. Le interruzioni che possono invece essere abilitate per mezzo di bit dedicati e un bit globale, sono dette mascherabili, ovvero potrebbero essere ignorate.

Cosa avviene quando un evento cerca d'interrompere l'esecuzione del programma? L'Atmega 2560 risponde in 5 cicli di clock prima di iniziare ad eseguire il programma (istruzioni) associate all'interruzione. Prima di fare questo deve essere salvato l'indirizzo dell'istruzione successiva da eseguire, infatti prima di eseguire l'interruzione viene terminata l'operazione corrente. L'indirizzo viene memorizzato nella memoria RAM nella

zona detta *Stack*. L'indirizzo di tale zona di memoria inizia dal valore impostato nello *Stack Pointer*. Molte MCU, oltre a salvare l'indirizzo di memoria programma dover tornare ad eseguire il programma, salvano anche il valore dello *Status Register* (Registro di Stato).Questo non viene fatto dall'Atmega2560, per cui se necessario, bisogna salvare e ripristinare lo *Status Register* via software. Quando l'interruzione viene accettata, la CPU disabilita il bit globale delle interruzione I-bit presente nello *Status Register*, al fine di evitare che possano avvenire interrupt annidati (*Nested Interrupt*). Questo non significa che altri eventi di interruzione saranno persi, infatti tutti gli eventi accettati possiedono un proprio flag che tiene memoria di un evento, il quale una volta terminate le istruzioni associate al programma di gestione delle interruzioni permettono di avviare una nuova sequenza d'interruzione. Da questo si capisce che eventi multipli di uno stesso tipo saranno contati solo una volta, infatti il flag non è un contatore. Al termine di una sequenza di interruzione l'indirizzo da cui riprendere l'esecuzione normale viene ripristinato nel *Program Counter* e le interruzioni sono riabilitate per mezzo del bit I-bit dello *Status Register*.

Vector No.	Program Address(2)	Source	Interrupt Definition
1	$0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, Watchdog Reset, and JTAG AVR Reset
2	$0002	INT0	External Interrupt Request 0
3	$0004	INT1	External Interrupt Request 1
4	$0006	INT2	External Interrupt Request 2
5	$0008	INT3	External Interrupt Request 3
6	$000A	INT4	External Interrupt Request 4
7	$000C	INT5	External Interrupt Request 5
8	$000E	INT6	External Interrupt Request 6
9	$0010	INT7	External Interrupt Request 7
10	$0012	PCINT0	Pin Change Interrupt Request 0
11	$0014	PCINT1	Pin Change Interrupt Request 1
12	$0016(3)	PCINT2	Pin Change Interrupt Request 2
13	$0018	WDT	Watchdog Time-out Interrupt
14	$001A	TIMER2 COMPA	Timer/Counter2 Compare Match A
15	$001C	TIMER2 COMPB	Timer/Counter2 Compare Match B
16	$001E	TIMER2 OVF	Timer/Counter2 Overflow
17	$0020	TIMER1 CAPT	Timer/Counter1 Capture Event
18	$0022	TIMER1 COMPA	Timer/Counter1 Compare Match A
19	$0024	TIMER1 COMPB	Timer/Counter1 Compare Match B
20	$0026	TIMER1 COMPC	Timer/Counter1 Compare Match C
21	$0028	TIMER1 OVF	Timer/Counter1 Overflow
22	$002A	TIMER0 COMPA	Timer/Counter0 Compare Match A
23	$002C	TIMER0 COMPB	Timer/Counter0 Compare match B
24	$002E	TIMER0 OVF	Timer/Counter0 Overflow
25	$0030	SPI, STC	SPI Serial Transfer Complete
26	$0032	USART0 RX	USART0 Rx Complete
27	$0034	USART0 UDRE	USART0 Data Register Empty
28	$0036	USART0 TX	USART0 Tx Complete
29	$0038	ANALOG COMP	Analog Comparator
30	$003A	ADC	ADC Conversion Complete

Tabella 1: *Reset e Interrupt Vector.*

Vector No.	Program Address(2)	Source	Interrupt Definition
31	$003C	EE READY	EEPROM Ready
32	$003E	TIMER3 CAPT	Timer/Counter3 Capture Event
33	$0040	TIMER3 COMPA	Timer/Counter3 Compare Match A
34	$0042	TIMER3 COMPB	Timer/Counter3 Compare Match B
35	$0044	TIMER3 COMPC	Timer/Counter3 Compare Match C
36	$0046	TIMER3 OVF	Timer/Counter3 Overflow
37	$0048	USART1 RX	USART1 Rx Complete
38	$004A	USART1 UDRE	USART1 Data Register Empty
39	$004C	USART1 TX	USART1 Tx Complete
40	$004E	TWI	2-wire Serial Interface
41	$0050	SPM READY	Store Program Memory Ready
42	$0052(3)	TIMER4 CAPT	Timer/Counter4 Capture Event
43	$0054	TIMER4 COMPA	Timer/Counter4 Compare Match A
44	$0056	TIMER4 COMPB	Timer/Counter4 Compare Match B
45	$0058	TIMER4 COMPC	Timer/Counter4 Compare Match C
46	$005A	TIMER4 OVF	Timer/Counter4 Overflow
47	$005C(3)	TIMER5 CAPT	Timer/Counter5 Capture Event
48	$005E	TIMER5 COMPA	Timer/Counter5 Compare Match A
49	$0060	TIMER5 COMPB	Timer/Counter5 Compare Match B
50	$0062	TIMER5 COMPC	Timer/Counter5 Compare Match C
51	$0064	TIMER5 OVF	Timer/Counter5 Overflow
52	$0066(3)	USART2 RX	USART2 Rx Complete
53	$0068(3)	USART2 UDRE	USART2 Data Register Empty
54	$006A(3)	USART2 TX	USART2 Tx Complete
55	$006C(3)	USART3 RX	USART3 Rx Complete
56	$006E(3)	USART3 UDRE	USART3 Data Register Empty
57	$0070(3)	USART3 TX	USART3 Tx Complete

Notes: 1. When the BOOTRST Fuse is programmed, the device will jump to the Boot Loader address at reset, see "Memory Programming" on page 325.
2. When the IVSEL bit in MCUCR is set, Interrupt Vectors will be moved to the start of the Boot Flash Section. The address of each Interrupt Vector will then be the address in this table added to the start address of the Boot Flash Section.
3. Only available in ATmega640/1280/2560.

Tabella 2: *Reset e Interrupt Vector.*

Come detto il vettore 1 (RESET) comanda su tutti gli altri, essendo chiaramente il primo, e come tutti gli altri che lo seguono comandano sui successivi in scala gerarchica. Nelle tabelle, oltre il numero del vettore, indirizzo esadecimale (*address*), sono mostrate le relative periferiche ed eventi ai quali è associato il vettore di interruzione. In fase di scrittura di un programma, ad ognuna di queste sorgenti viene associato un *handler* ISR, ovvero una routine ISR (*Interrupt Service Routine*), di cui l'utilizzatore può disporre per scrivere il programma da eseguire per la gestione dell'interruzione ovvero dell'evento che l'ha generata.
La sintassi per scrivere un ISR è praticamente la stessa per ogni vettore. Nel caso di un ISR si deve scrivere:

```
ISR (sorgente_vect){

 // codice da eseguire

}
```

Per esempio per l'interrupt esterno INT0 (vettore n° 2) si scrive:

```
ISR (INT0_vect){

 // codice da eseguire
}
```

per l'interrupt sull'ADC (vettore n° 30) si scrive:

```
ISR (ADC_vect){

 // codice da eseguire
}
```

per l'interrupt interno, ISR sul bit TOV0 del timer/counter 0 (vettore n° 24) si scrive:

```
ISR (TIMER0_OVF_vect){

 // codice da eseguire
}
```

per l'interrupt interno, ISR sull'USART0 in RX (vettore n° 26) si scrive:

```
ISR (USART0_RX_vect){

 // codice da eseguire
}
```

Si tenga sempre presente che per innescare o meno un evento interrupt per qualunque periferica si usi, oltre ai bit di abilitazione richiesti nei vari registri secondo il caso, è necessario includere sempre nel `main` del programma la libreria `<avr/interrupt.h>` ed abilitare l'interrupt globale, ovvero il bit 7 (I-bit) del Registro di Stato SREG, secondo le seguenti istruzioni:

```
// Interrupt globale abilitato
// l'istruzione equivalente in assembler è sei();
SREG |= 0x80;

// Interrupt globale disabilitato
// l'istruzione equivalente in assembler è cli();
SREG &= 0x7F;
```

Eventi d'interruzione esterni

Gli interrupt esterni, per un totale di 32, sono innescati dai pin INT0-INT7 (*External Interrupt Request*) e dai pin PCINT0-PCINT23 (*Pin Change Interrupt Request*) dell'Atmega 2560. Ognuno di questi pin, se abilitato, provoca un interrupt anche se è impostato come uscita (output), sia sul fronte di salita (*Rising Edge*) che di discesa (*Falling Edge*) oppure quando messo a basso livello (GND). Ad un occhio attento però non sfugge il fatto che nella tavola dei vettori della serie *Pin Change Interrupt Request* sono presenti, in ordine gerarchico, solo tre livelli PCINT0, PCINT1 e PCINT2. Il livello PCINT0 si occupa di innescare l'interrupt per tutti i pin PCINT0-PCINT7, il livello PCINT1 si occupa di innescare l'interrupt per tutti i pin PCINT8-PCINT15 ed il livello PCINT2 si occupa di innescare l'interrupt per tutti i pin PCINT16-PCINT23.

Vediamo i vari dettagli dei registri associali alle interruzioni da parte di eventi esterni alla MCU.

EICRA: External Interrupt Control Register A - INT0-INT3

Questo registro, inizializzato a 0 e riportato in Tabella 3, controlla il livello di sensibilità (*sense*) tramite il quale un interrupt innescato dai pin esterni INT0-INT3 viene rilevato dall'hardware come tale. Per ognuno di questi pin, il livello di sensibilità viene stabilito dai bit ISCn0 e ISCn1, riportati in Tabella 3, dove n è uguale al pin esterno. Per il pin INT0 il livello di sensibilità è definito dai bit ISC00 è ISC01, per il pin INT1 è definito dai bit ISC10 è ISC11 e così di seguito. Prendendo come esempio il bit INT0, si ha che con i bit ISC00 e ISC10 azzerati, l'interrupt richiesto avviene con livello logico basso ovvero GND. Quando solo il bit ISC00 è posto ad 1, avviene sia in fase di salita (*rising*) che in fase di discesa (*falling*). Con il solo bit ISC01 posto ad 1 avviene solo in fase di discesa e con entrambi i bit ISC01 e ISC00 impostati ad 1, avviene solo in fase di salita. Tali interruzioni sono di tipo asincrono. La modalità asincrona è dovuta al fatto che questi interrupt possono essere usati per "risvegliare" il sistema, qualora sia stato posto in stato di *sleep* o di inattività, ed in particolare il risveglio può essere richiesto in qualunque momento (maniera asincrona). Per poter attivare l'interrupt sui pin INT0-INT3 non basta solo impostare il fronte a cui saranno sensibili, ma bisogna impostare ad 1, il bit corrispondente nel registro EIMSK e il bit I-bit del registro SREG.

Bit	7	6	5	4	3	2	1	0	
(0x69)	ISC31	ISC30	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabella 3: *EICRA - External Interrupt Control Register A.*

EICRB: External Interrupt Control Register B - INT4-INT7

Questo registro, inizializzato a 0 e riportato in Tabella 4, controlla il livello di sensibilità (*sense*) tramite il quale un interrupt innescato dai pin esterni INT4-INT7 viene rilevato dall'hardware come tale. Per ognuno di questi pin, il livello di sensibilità viene stabilito dai bit ISCn0 e ISCn1, riportati in Tabella 4, dove n è uguale al pin esterno. Per esempio, il livello di sensibilità per il pin INT4 è definito dai bit ISC40 è ISC41, per il pin INT5 è definito dai bit ISC50 è ISC51 e così di seguito. Prendendo come esempio il bit INT4, si ha che con i bit ISC40 e ISC41 azzerati, l'interrupt richiesto avviene con livello logico

basso ovvero GND. Quando solo il bit ISC40 è posto ad 1, avviene sia in fase di salita (*rising*) che in fase di discesa (*falling*). Con il solo bit ISC41 posto ad 1 avviene solo in fase di discesa e con entrambi i bit ISC41 e ISC00 impostati ad 1, avviene solo in fase di salita.

Bit	7	6	5	4	3	2	1	0	
(0x6A)	ISC71	ISC70	ISC61	ISC60	ISC51	ISC50	ISC41	ISC40	EICRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabella 4: *EICRB - External Interrupt Control Register B.*

Per poter attivare l'interrupt sui pin INT4-INT7 non basta solo impostare il fronte a cui saranno sensibili, ma bisogna impostare ad 1, il bit corrispondente nel registro EIMSK e il bit I-bit del registro SREG.

EIMSK: External Interrupt Mask Register

Questo registro permette di abilitare l'interruzione sui pin INT0-INT7, permettendo ad eventi esterni alla MCU di interrompere il flusso del programma. Quando i bit 0-7 di questo registro, inizializzati a 0, sono posti ad 1 insieme al bit I-bit dello *Status Register* SREG viene abilitato l'interrupt sui pin esterni INT0-INT7, mentre i registri EICRA-EICRB, come detto, gestiscono la soglia del suo rilevamento in base all'impostazione dei bit ISCn relativi agli stessi bit. In Tabella 5 è mostrato il dettaglio del registro.

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0	EIMSK
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabella 5: *EIMSK - External Interrupt Mask Register.*

EIFR: External Interrupt Flag Register

Questo registro è inizialmente inizializzato a 0. Al verificarsi di un cambiamento logico o fronte sui pin INT0-INT7 (a seconda delle relative impostazioni delle interruzioni) il relativo bit INTF0-INTF7 viene posto ad 1. Se le interruzioni sui pin sono abilitate avviene il salto all'ISR. Il flag, ovvero bit relativo al pin che ha generato l'interruzione viene automaticamente posto a 0 una volta eseguita l'ISR. In alternativa si può scrivere 1 nel bit relativo. Ovvero il processo di scrittura resetta il bit. La procedura manuale di *Reset* del flag è buona norma farla ogni qual volta si attivi l'interruzione per la prima volta. Ovvero prima di attivare le interruzioni si resetta il flag d'interesse al fine di ignorare eventuali eventi pendenti che dovessero aver preceduto l'abilitazione delle interruzioni. Questa procedura è bene seguirla anche per altri flag delle interruzioni.

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	INTF7	INTF6	INTF5	INTF4	INTF3	INTF2	INTF1	IINTF0	EIFR
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabella 6: *EIFR - External Interrupt Flag Register.*

PCICR: Pin Change Interrupt Control Register

Questo registro, i cui dettagli sono mostrati in Tabella 7, è inizializzato a 0 e riguarda solo i tre bit PCIE0-PCIE2.

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabella 7: *PCICR - Pin Change Interrupt Control Register.*

PCIE0: Pin Change Interrupt Enable 0

Quando questo bit, inizialmente inizializzato a 0, è posto ad 1 ed anche il bit I-bit di SREG vale 1, viene abilitato l'interrupt sul pin PCINT0. Ogni cambio logico sui pin PCINT0-PCINT7, se abilitati individualmente nel registro PCMSK0, causa un evento di interrupt. La relativa routine ISR per ciascuno di questi pin viene eseguita sempre dal vettore PCINT0.

PCIE1: Pin Change Interrupt Enable 1

Quando questo bit, inizialmente inizializzato a 0, è posto ad 1 ed anche il bit I-bit di SREG vale ad 1, viene abilitato l'interrupt sul pin PCINT1. Ogni cambio logico sui pin PCINT8-PCINT15, se abilitati individualmente nel registro PCMSK1, causa un evento d'interrupt. La relativa routine ISR per ciascuno di questi pin viene eseguita sempre dal vettore PCINT1.

PCIE2: Pin Change Interrupt Enable 2

Quando questo bit, inizialmente inizializzato a 0, è posto ad 1 ed anche il bit I-bit di SREG vale ad 1, viene abilitato l'interrupt sul pin PCINT2. Ogni cambio logico sui pin PCINT16 - PCINT23, se abilitati individualmente nel registro PCMSK2, causa un evento d'interrupt. La relativa routine ISR per ciascuno di questi pin viene eseguita sempre dal vettore PCINT2.

PCIFR: Pin Change Interrupt Flag Register

Questo registro, di cui è riportato il dettaglio in Tabella 8, è inizializzato a 0 e riguarda solo i tre bit PCIF0-PCIF2. Il suo contenuto è dato dai relativi flag delle interruzioni dei rispettivi pin PCINTx.

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	–	–	–	–	–	PCIF2	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabella 8: *PCIFR - Pin Change Interrupt Flag Register.*

In particolare quando un livello logico sui pin PCINT0-PINT7 viene cambiato, il bit PCIF0 viene posto a 1. Se le relative interruzioni, ovvero il bit PCIE0 e I-bit del registro SREG sono posti ad 1, viene generata una interruzione e il programma salta all'ISR. Il relativo flag viene automaticamente resettato all'esecuzione dell'ISR. I flag PCIF1 e PCIF2 sono associati ai pin abilitati per mezzo dei bit PCIE1 e PCIE2.

PCMSK0: Pin Change Mask Register 0

Questo registro è inizializzato 0. Quando i bit 0-7 di questo registro sono posti ad 1 vengono selezionati (abilitati) i *pin change* PCINT0-PCINT7. Se nel registro PCICR viene posto ad 1 il bit PCIE0, insieme al bit I-bit del registro SREG, l'interrupt viene abilitato. In Tabella 9 è riportato il dettaglio del registro:

Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabella 9: *PCMSK0 – Pin Change Mask Register 0.*

PCMSK1: Pin Change Mask Register 1

Questo registro è inizializzato 0. Quando i bit 0-7 di questo registro sono posti ad 1 vengono selezionati (abilitati) i *pin change* PCINT8-PCINT15. Se nel registro PCICR viene posto ad 1 il bit PCIE1, insieme al bit I-bit del registro SREG, l'interrupt viene abilitato. In Tabella 10 è riportato il dettaglio del registro:

Bit	7	6	5	4	3	2	1	0	
(0x6C)	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabella 10: *PCMSK1 – Pin Change Mask Register 1.*

PCMSK2: Pin Change Mask Register 2

Questo registro è inizializzato 0. Quando i bit 0-7 di questo registro sono posti ad 1 vengono selezionati (abilitati) i *pin change* PCINT16-PCINT23. Se nel registro PCICR viene posto ad 1 il bit PCIE2, insieme al bit I-bit del registro SREG, l'interrupt viene abilitato. In Tabella 11 è riportato il dettaglio del registro:

Bit	7	6	5	4	3	2	1	0	
(0x6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Tabella 11: *PCMSK2 – Pin Change Mask Register 2.*

Esempi d'impostazione di un interrupt esterno

Dopo questa presentazione dei registri, in questo paragrafo verrà descritto sinteticamente come impostarli per abilitare un evento interrupt ed intercettarlo.

Esempio d'impostazione interrupt sul pin INT0

Problema
Si deve abilitare un interrupt asincrono sul pin INT0 ed intercettarlo sul fronte di discesa.

Questo pin fa parte della serie INT0-INT3 del registro EICRA (*External Interrupt Control Register A*, Tabella 3) i cui bit di *sense* corrispondenti sono ISC00 e ISC10. Dovendo intercettare l'interrupt sul fronte di discesa si dovranno scrivere inizialmente ad 1 questi bit:

```
EICRA |= (1<<ISC00) | (1<<ISC10);
```

Poi si dovrà abilitare l'interrupt dello stesso bit nel registro EIMSK (*External Interrupt Mask Register*, Tabella 5):

```
EIMSK |= (1<<INT0);
```

Si dovrà anche resettare il flag corrispondente nel registro EIFR (*External Interrupt Flag Register*, Tabella 6) al fine di cancellare eventuali eventi non d'interesse su INT0:

```
EIFR |= (1<<INTF0);
```

Infine si dovrà abilitare l'interrupt globale I-bit (bit 7) nel registro SREG:

```
SREG |=0x80;  // oppure (1<<7);  o l'equivalente assembler sei();
```

Esempio di impostazione interrupt sul pin change PCINT5

Problema
Si deve abilitare un interrupt sul pin change PCINT5.

Questo pin appartiene alla serie PCINT0-PCINT7 ed è gestito, in base alla tavola dei vettori in Tabella 1), dal bit PCIE0 (*Pin Change Interrupt Enable 0*) presente nel registro PCICR (*Pin Change Interrupt Control Register*, Tabella 7). Si dovrà innanzitutto abilitare l'interrupt, scrivendo ad 1, il pin PCIE0:

```
PCICR |= (1<<PCIE0);
```

Poi si dovrà resettare il relativo flag, scrivendo ad 1, il bit corrispondente nel registro PCIF0 (*Pin Change Interrupt Flag 0*), presente nel registro PCIFR, Tabella 8). Questo

permette di resettare eventuali eventi pendenti non d'interesse.

```
PCIFR |= (1<<PCIF0);
```

In seguito si dovrà abilitare, scrivendo ad 1, il bit PCINT0 nel registro PCMSK0 (*Pin Change Mask Register 0*, Tabella 9):

```
PCMSK0 |= (1<<PCINT0):
```

Infine si dovrà abilitare l'interrupt globale I-bit (bit 7) nel registro SREG:

```
SREG |=0x80;  // oppure (1<<7);  o l'equivalente assembler sei();
```

si ricorda in ultimo che in tutti i programmi in cui si fa uso delle interruzioni bisogna includere il file:

```
#include <avr/interrupt.h>
```

Si lascia al lettore l'esercizio di cambiare l'esempio basato sul pulsante e accensione del LED e convertirlo dall'utilizzo del *polling* (lettura continua dello stato del pulsante), ad interruzione.

Indice Alfabetico

A
ADC....7
E
EICRA....8 e seg., 12
EICRB....9
EIFR....12
EIMSK....8 e seg., 12
External Interrupt Control Register....8
External Interrupt Request....8
I
INT0....8
Interrupt Service Routine....6
ISCn....9
ISR....6, 9
M
MCU Control Register....4
MCUCR....4
N
Nested Interrupt....5
P
PCICR....12
PCIE0....10, 12
PCIE2....10
PCINT0....8, 10, 13
PCINT1....10
PCINT2....10
PCMSK0....10, 13
PCMSK1....10
PCMSK2....10
Pin Change Interrupt Request....8
R
Registro di Stato....5, 7
Reset e Interrupt Vector....4
Rising Edge....8
S
sense....8
SREG....7 e segg.
Stack....5
Stack Pointer....5
Status Register....5, 9
T
TOV0....7
V
Vector Number....4

History

Data	Versione	Autore	Revisione	Descrizione Cambiamento
23/05/20	1.2	Marcello Pinna	Mauro Laurenti	Cambiati bit ICS in ISC.
20/04/20	1.1	Marcello Pinna	Mauro Laurenti	Pagina 7, aggiornato reset bit Interrupt con SREG &= 0x7F;
18/04/20	1.0	Marcello Pinna	Mauro Laurenti	Versione Originale.