Interrupts

main code or subroutine code

mov.w #3, R5

If hardware requests an interrupt while this instruction is being executed then this instruction is completed first then execution starts from the ISR

Interrupt Service Routine

label → yourISR: ← your code related to the interrupt

"function" called by the hardware at unpredictable times

reti

return from interrupt execution starts from here

interrupts are used to handle:

Urgent tasks
Slow human input
Waking the CPU from sleep

etc.

Interrupts can be requested by:

peripherals:

e.g ADC10 (Analog to Digital Converter)
when an input analog voltage has been sampled and is ready for your program to read

Timer A

Every time the counter register reaches its maximum value

GPIO

whenever the input signal on a port pin has a rising edge or a falling edge (provided that pin has been configured as an input pin with interrupt enabled)

Every interrupt type has a flag which is set when the conditions for that interrupt occur.

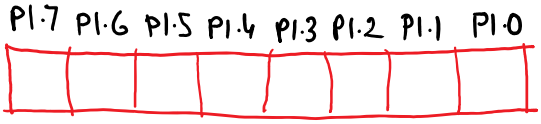
Interrupt flags for port1 GPIO interrupts for example:

flags →

P1IFG0 ⟷ P1.0
P1IFG1 ⟷ P1.1
P1IFG2 ⟷ P1.2
⋮
P1IFG7 ⟷ P1.7

Each flag has a corresponding interrupt enable bit which must be set to enable that interrupt; e.g

P1IE →

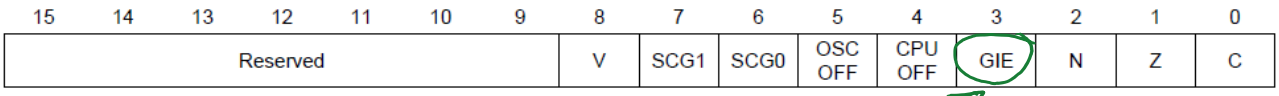


For hardware to request an interrupt the corresponding interrupt bit must be set.

Two types of interrupts

Maskable interrupts:

Status Register (SR)



General Interrupt Enable

maskable interrupts are ignored if the GIE bit is not set.

To enable maskable interrupts:

Set the interrupt enable bit for that interrupt

+

enable GIE in the SR

(bis.w #GIE, SR)

Non-Maskable Interrupts

GIE bit has no effect on non-maskable interrupts

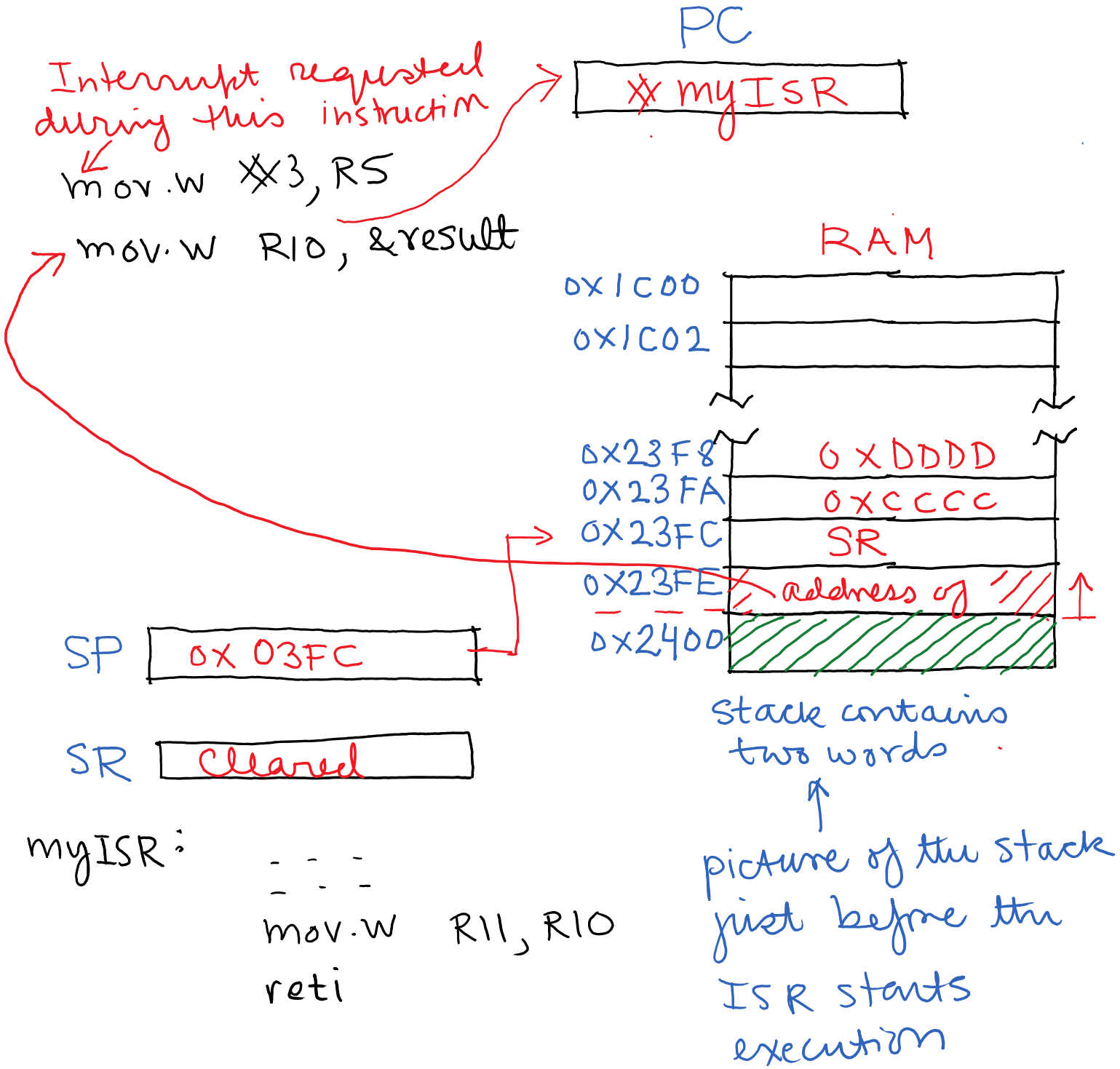
(their interrupt bit must still be enabled, which is not set by default)

see your text book page 188

When an interrupt is requested then:

* Any currently executing instruction is completed
* The PC, which points to the next instruction is pushed on to the stack
* The Status register (SR) is pushed on to the stack
* The interrupt with the highest priority is selected if multiple interrupts are waiting for service
* The SR is cleared, further maskable interrupts are disabled since GIE bit is cleared. All low power modes are terminated.
* address of the corresponding ISR is loaded into the PC.
* ISR starts execution

Stack picture



When reti is executed

* The SR pops from the stack. All previous settings of GIE and Low power mode control bits are in effect again, i.e, maskable interrupts are enabled again.

* PC pops from the stack and execution resumes from the point where it was interrupted. If the CPU was asleep then it goes back to sleep.

Linker File in CCS

lnk_msp430fr6989.cmd | main.asm

Blinky/lnk_msp430fr6989.cmd

```
62    INFOC                 : origin = 0x1880, length = 0x0080
63    IN                    : origin = 0x1800, length = 0x0080
64    FRAM                  : origin = 0x4400, length = 0xBB80
65    FRAM2                 : origin = 0x10000,length = 0x14000
66    JTAGSIGNATURE         : origin = 0xFF80, length = 0x0004, fill = 0xFFFF
67    BSLSIGNATURE          : origin = 0xFF84, length = 0x0004, fill = 0xFFFF
68    IPESIGNATURE          : origin = 0xFF88, length = 0x0008, fill = 0xFFFF
69    INT00                 : origin = 0xFF90, length = 0x0002
70    INT01                 : origin = 0xFF92, length = 0x0002
71    INT02                 : origin = 0xFF94, length = 0x0002
72    INT03                 : origin = 0xFF96, length = 0x0002
73    INT04                 : origin = 0xFF98, length = 0x0002
74    INT05                 : origin = 0xFF9A, length = 0x0002
75    INT06                 : origin = 0xFF9C, length = 0x0002
76    INT07                 : origin = 0xFF9E, length = 0x0002
77    INT08                 : origin = 0xFFA0, length = 0x0002
78    INT09                 : origin = 0xFFA2, length = 0x0002
79    INT10                 : origin = 0xFFA4, length = 0x0002
80    INT11                 : origin = 0xFFA6, length = 0x0002
81    INT12                 : origin = 0xFFA8, length = 0x0002
82    INT13                 : origin = 0xFFAA, length = 0x0002
83    INT14                 : origin = 0xFFAC, length = 0x0002
84    INT15                 : origin = 0xFFAE, length = 0x0002
85    INT16                 : origin = 0xFFB0, length = 0x0002
86    INT17                 : origin = 0xFFB2, length = 0x0002
87    INT18                 : origin = 0xFFB4, length = 0x0002
88    INT19                 : origin = 0xFFB6, length = 0x0002
89    INT20                 : origin = 0xFFB8, length = 0x0002
90    INT21                 : origin = 0xFFBA, length = 0x0002
91    INT22                 : origin = 0xFFBC, length = 0x0002
92    INT23                 : origin = 0xFFBE, length = 0x0002
93    INT24                 : origin = 0xFFC0, length = 0x0002
94    INT25                 : origin = 0xFFC2, length = 0x0002
95    INT26                 : origin = 0xFFC4, length = 0x0002
96    INT27                 : origin = 0xFFC6, length = 0x0002
97    INT28                 : origin = 0xFFC8, length = 0x0002
98    INT29                 : origin = 0xFFCA, length = 0x0002
99    INT30                 : origin = 0xFFCC, length = 0x0002   PORT4
100   INT31                 : origin = 0xFFCE, length = 0x0002   PORT3
101   INT32                 : origin = 0xFFD0, length = 0x0002
102   INT33                 : origin = 0xFFD2, length = 0x0002
103   INT34                 : origin = 0xFFD4, length = 0x0002   PORT2
104   INT35                 : origin = 0xFFD6, length = 0x0002
105   INT36                 : origin = 0xFFD8, length = 0x0002
106   INT37                 : origin = 0xFFDA, length = 0x0002   PORT1
107   INT38                 : origin = 0xFFDC, length = 0x0002
108   INT39                 : origin = 0xFFDE, length = 0x0002
```

Device Specific Data Sheet SLAS789D

Table 6-4. Interrupt Sources, Flags, and Vectors (continued)

INTERRUPT SOURCE	INTERRUPT FLAG	SYSTEM INTERRUPT	WORD ADDRESS	PRIORITY
eUSCI_B1 receive or transmit)	UCB1IFG: UCRXIFG, UCTXIFG (SPI mode) UCB1IFG: UCALIFG, UCNACKIFG, UCSTTIFG, UCSTPIFG, UCRXIFG0, UCTXIFG0, UCRXIFG1, UCTXIFG1, UCRXIFG2, UCTXIFG2, UCRXIFG3, UCTXIFG3, UCCNTIFG, UCBIT9IFG (I^2C mode) (UCB1IV)(1)	Maskable	0FFE2h	
DMA	DMA0CTL.DMAIFG, DMA1CTL.DMAIFG, DMA2CTL.DMAIFG (DMAIV)(1)	Maskable	0FFE0h	
Timer_A TA1	TA1CCR0.CCIFG	Maskable	0FFDEh	
Timer_A TA1	TA1CCR1.CCIFG to TA1CCR2.CCIFG, TA1CTL.TAIFG (TA1IV)(1)	Maskable	0FFDCh	
I/O Port P1	P1IFG.0 to P1IFG.7 (P1IV)(1)	Maskable	0FFDAh	
Timer_A TA2	TA2CCR0.CCIFG	Maskable	0FFD8h	
Timer_A TA2	TA2CCR1.CCIFG TA2CTL.TAIFG (TA2IV)(1)	Maskable	0FFD6h	
I/O Port P2	P2IFG.0 to P2IFG.7 (P2IV) (1)	Maskable	0FFD4h	
Timer_A TA3	TA3CCR0.CCIFG	Maskable	0FFD2h	
Timer_A TA3	TA3CCR1.CCIFG TA3CTL.TAIFG (TA3IV)(1)	Maskable	0FFD0h	
I/O Port P3	P3IFG.0 to P3IFG.7 (P3IV) (1)	Maskable	0FFCEh	
I/O Port P4	P4IFG.0 to P4IFG.7 (P4IV) (1)	Maskable	0FFCCh	
LCD_C (Reserved on MSP430FR5xxx)	LCD_C interrupt flags (LCDCIV) (1)	Maskable	0FFCAh	
RTC_C	RTCRDYIFG, RTCTEVIFG, RTCAIFG, RT0PSIFG, RT1PSIFG, RTCOFIFG (RTCIV) (1)	Maskable	0FFC8h	

Setting the interrupt vector

main program and subroutines

```
PORT1_ISR:
            .
            .
            .
            clr.b   &P1IFG              ; multi sourced interrupt, therefor you must
                                        ;clear the interrupt flag

            reti                        ; return from interrupt

;-------------------------------------------------------------------------
;   Interrupt Vectors
;-------------------------------------------------------------------------
            .sect       ".int37"
            .short    PORT1_ISR

            .sect   ".reset"
            .short  RESET
```