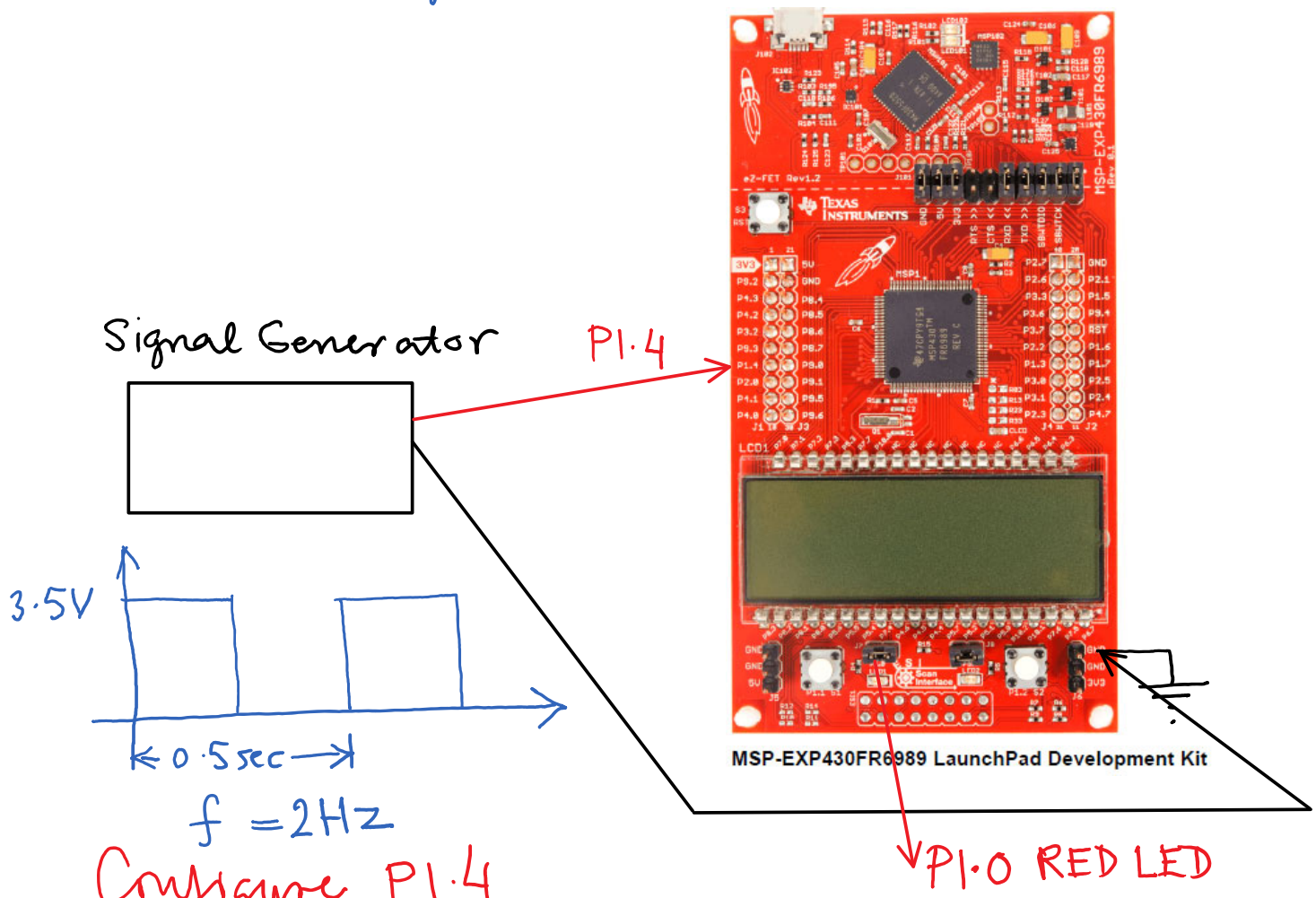
Interrupt Based Blinkies

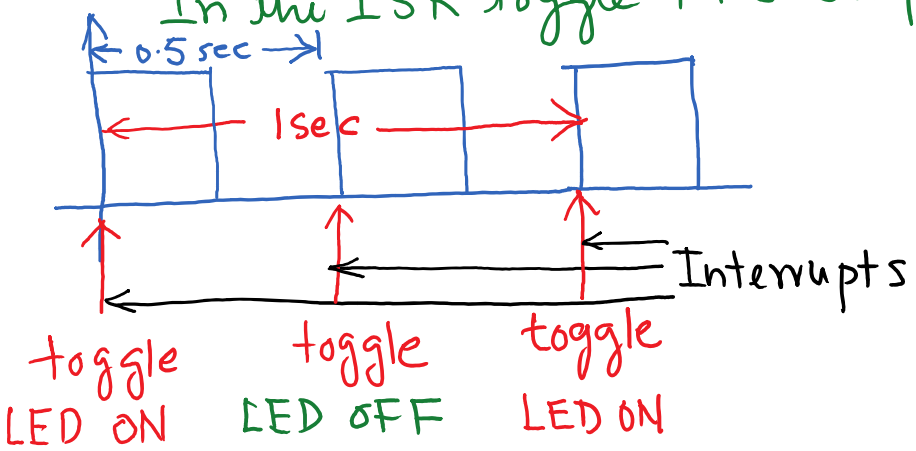


Configure P1.4

input, no resistor, interrupt enabled, rising edge interrupt

Configure P1.0 → output pin

In the ISR toggle P1.0 output



LED will blink at the rate of 1 blink/second

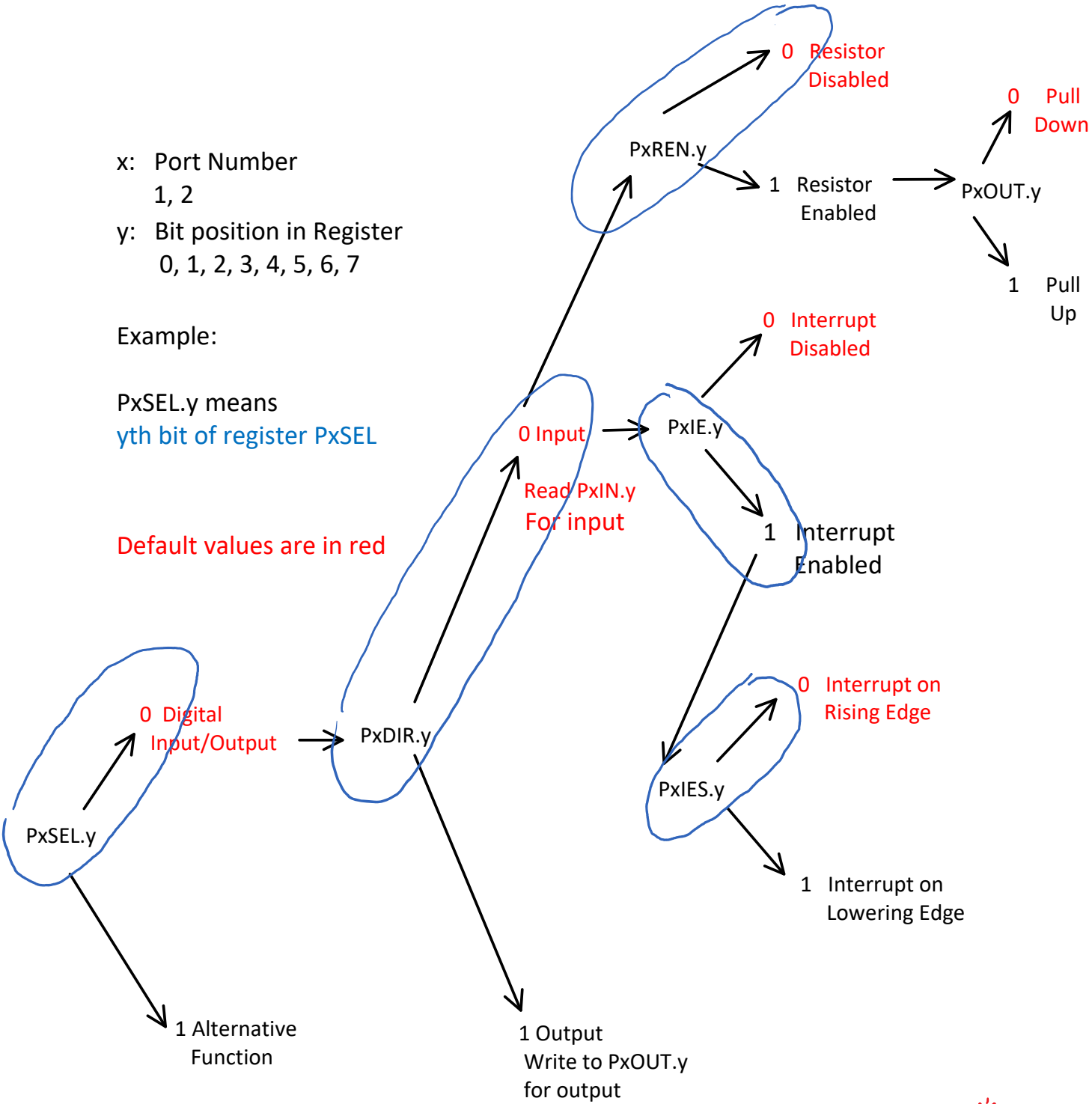
Configuring P1.4 for interrupts

x: Port Number
1, 2
y: Bit position in Register
0, 1, 2, 3, 4, 5, 6, 7

Example:

PxSEL.y means
yth bit of register PxSEL

Default values are in red



bic.b #BIT4, &P1SEL*
bic.b #BIT4, &P1REN*
bic.b #BIT4, &P1IES*

bic.b #BIT4, &PDIR*
bis.b #BIT4, &P1IE
* (default)

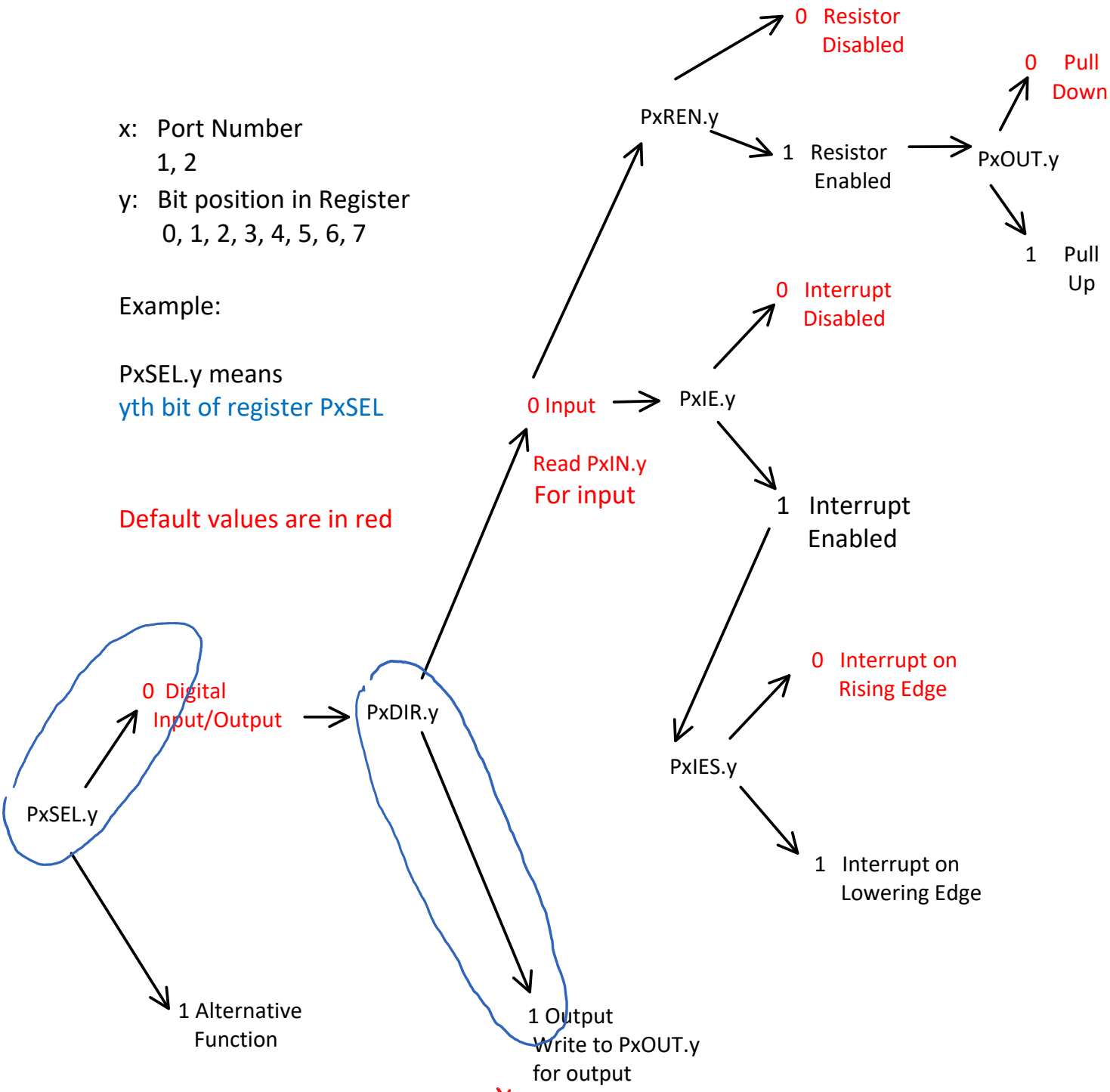
Configuring P1.0 (RED LED)

x: Port Number
1, 2
y: Bit position in Register
0, 1, 2, 3, 4, 5, 6, 7

Example:

PxSEL.y means
yth bit of register PxSEL

Default values are in red



bic.b #BIT0, &P1SEL* bis.b #BIT0, &P1DIR

* → default

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	

Linker File in CCS

lnk_msp430fr6989.cmd | main.asm

```
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
```

Blinky/lnk_msp430fr6989.cmd

Program: InterruptBadBlinky

```
                                .
                                .
                                .
            bis.b       #BIT0,      &P1OUT
            bis.b       #BIT0,      &P1DIR      ; Set P1.0 as output pin
                                                ; P1.0 connected to the red LED

            bis.b       #BIT4,      &P1IE       ; Enable P1.4 interrupts
            bic.w   #LOCKLPM5,&PM5CTL0          ; Disable the GPIO power-on default
                                                ; high-impedance mode

            nop                                 ; no operation
            bis.w       #GIE,       SR          ; Enable General Interrupts
            nop                                 ; no operation

loop:       jmp         loop                    ; loop and wait for interrupts
                                                ; bad Blinky bad!
                                                ; CPU should be sleeping

;------------------------------  ISR  ------------------------------------
PORT1_ISR:
            xor.b   #BIT0,      &P1OUT          ; Toggle P1.0
            bic.b   #BIT4,      &P1IFG          ; multi-sourced, therefore clear IF
            reti

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

read "Low power modes of operation, page 198, of your text book

Low Power Modes

CPU can go to sleep (in a low power mode and can be woken up by an interrupt

There are five Low Power Modes (LPMs)
We will look at only two

Active Mode: Fully Awake (CPU, all clocks and enable modes active)
Current consumed ≈ 300μA

LPM3: CPU, MCLK, SMCLK and DCO are disabled, only ACLK remains active.
current consumed ≈ 0.5μA

Status Register (SR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C

These 4 bits control the power modes

Status Register (SR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C
								0	0	0	0				

Active Mode

Header file defines:

```
#define LPM0                    (CPUOFF)
#define LPM1                    (SCG0+CPUOFF)
#define LPM2                    (SCG1+CPUOFF)
#define LPM3                    (SCG1+SCG0+CPUOFF)
#define LPM4                    (SCG1+SCG0+OSCOFF+CPUOFF)
```

Status Register (SR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Reserved							V	SCG1	SCG0	OSC OFF	CPU OFF	GIE	N	Z	C
								1	1	0	1				

LPM3 Mode

Set MCU in Low Power Mode 3:

```
BIS.W   #LPM3, SR
```

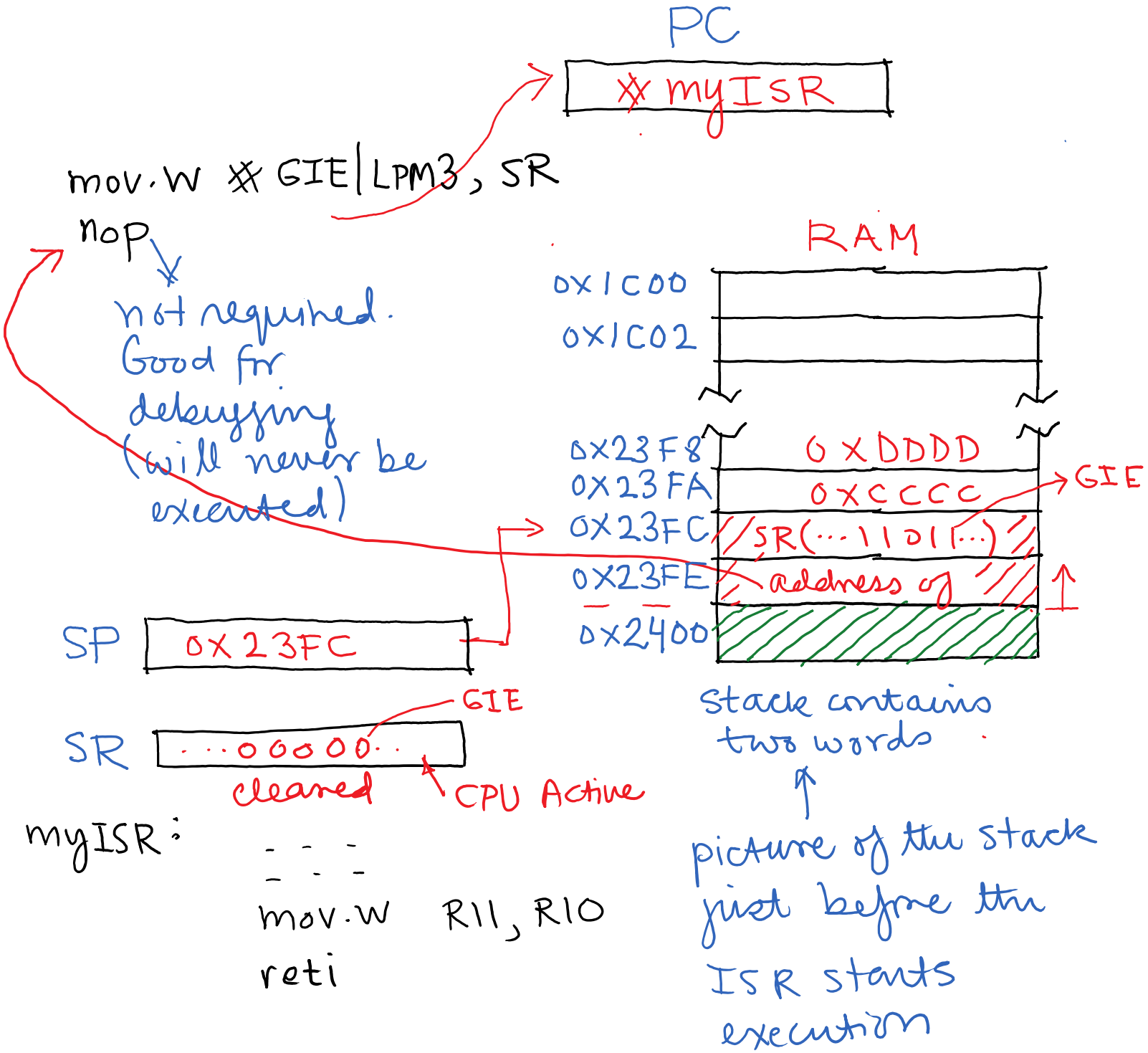
Enable general Interrupts & LPM3

```
BIS.W   #GIE|LPM3, SR
```

Program: InterruptGoodBlinky

```
                                .
                                .
                                .
            bis.b       #BIT0,      &P1OUT
            bis.b       #BIT0,      &P1DIR      ; Set P1.0 as output pin
                                                ; P1.0 connected to the red LED

            bis.b       #BIT4,      &P1IE       ; Enable P1.4 interrupts

           bic.w   #LOCKLPM5,&PM5CTL0      ; Disable the GPIO power-on default
                                           ; high-impedance mode

            nop                                 ; no operation
            bis.w       #GIE|LPM3,  SR          ; sleep and wait for interrupts
                                                ; Good Blinky!
            nop                                 ; no operation

;------------------------------  ISR  -------------------------------------
PORT1_ISR:
                xor.b   #BIT0,      &P1OUT      ; Toggle P1.0
                bic.b   #BIT4,      &P1IFG      ; multi-sourced, therefore clear IF
                reti

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

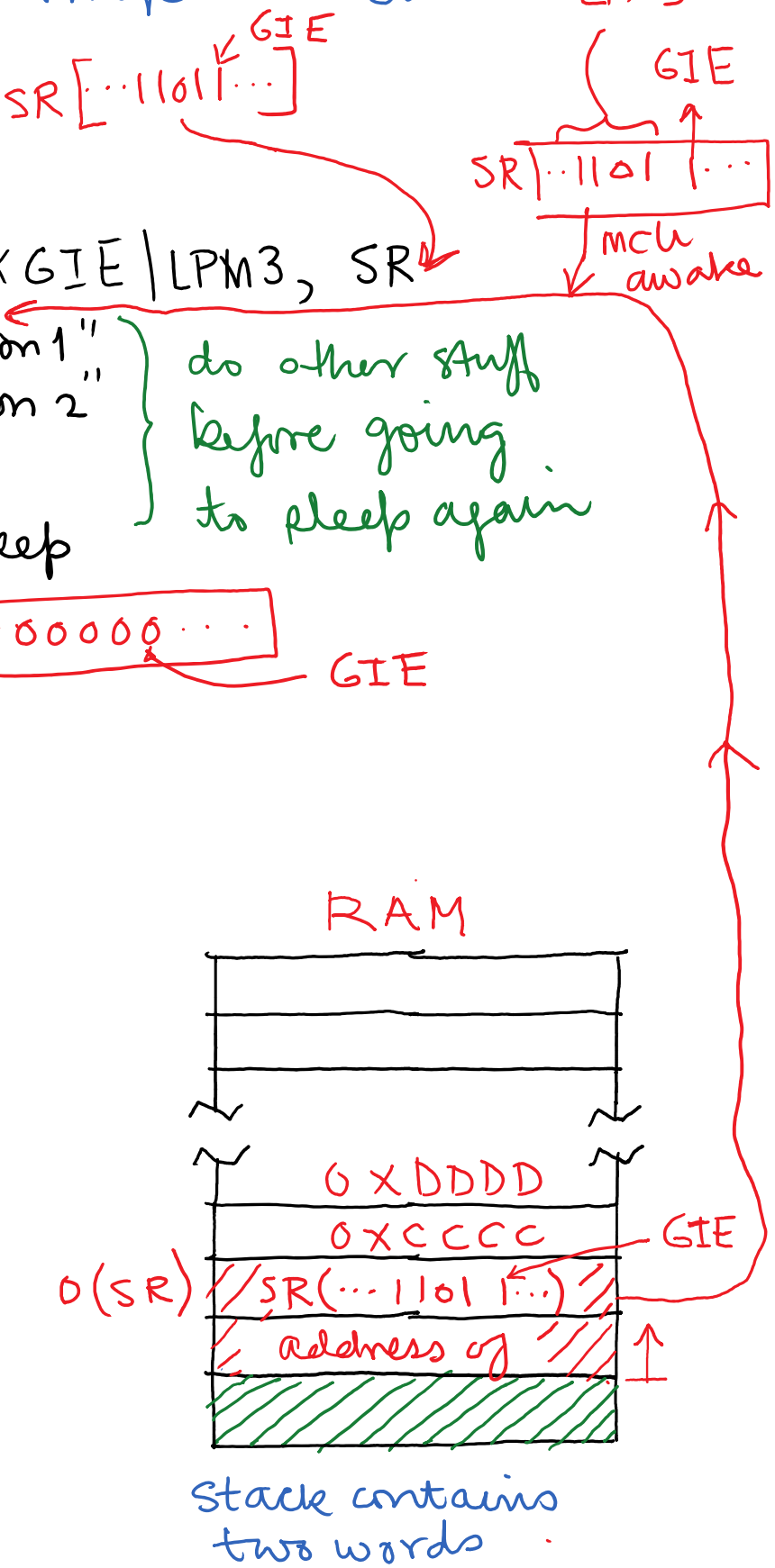
Stack picture



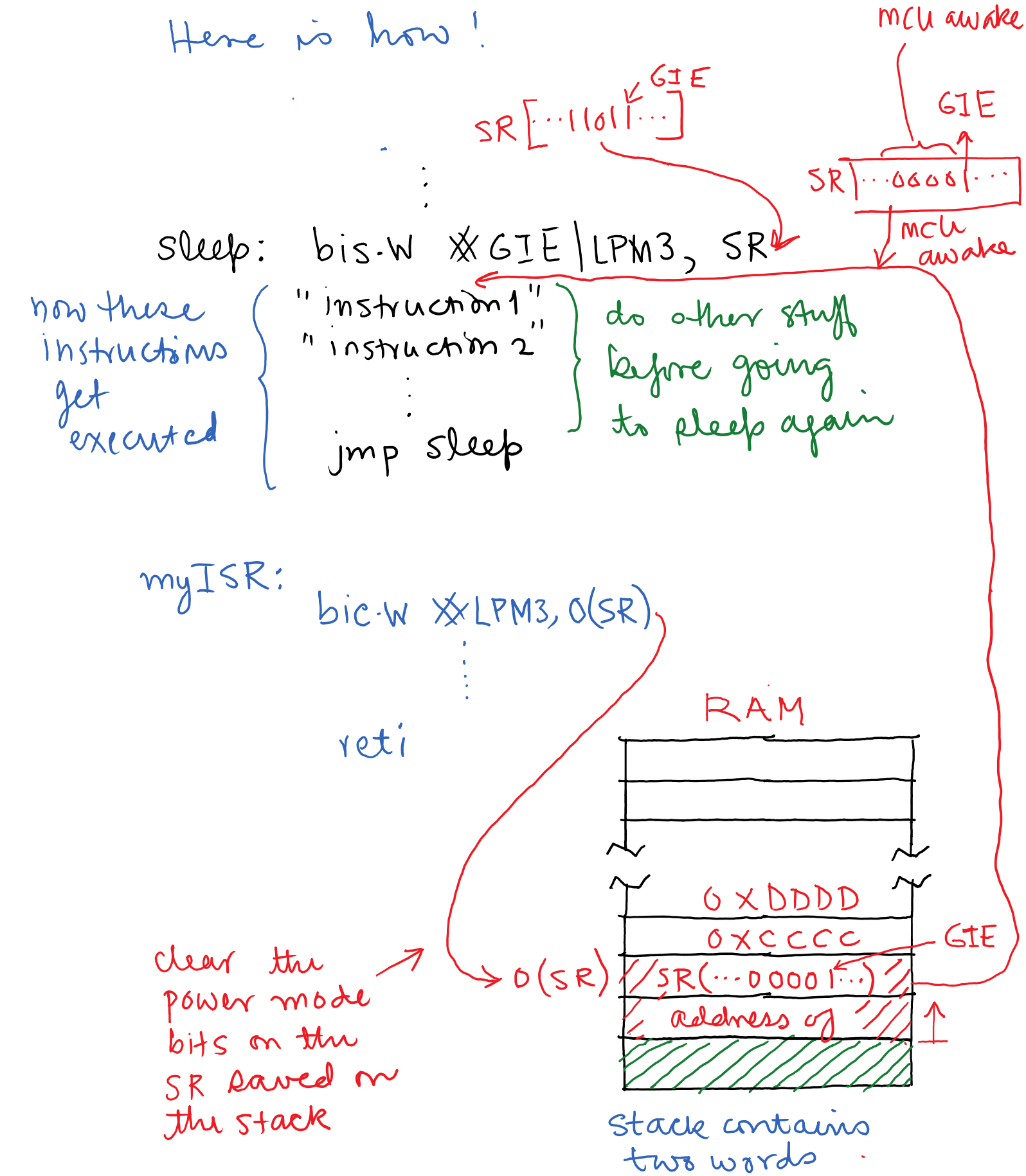
When reti is executed

* The SP 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, and the MCU goes back to sleep again

How do we accomplish the following ?



Here is how!


MCU awake
GIE
SR [···11011···]
GIE
SR [···00001···]
MCU awake
sleep: bis.w #GIE|LPM3, SR
now these instructions get executed
"instruction 1"
"instruction 2"
jmp sleep
do other stuff before going to sleep again
myISR:
bic.w #LPM3, 0(SR)
reti
RAM
0xDDDD
0xCCCC
GIE
0(SR)
SR(···00001···)
address of
clear the power mode bits on the SR saved on the stack
stack contains two words