Interrupts

Last updated 5/21/19

Interrupts

- Why Interrupts
  - I/O operations
    - Peripherals and external devices can let the processor know they are ready vs. the processor polling each device
  - Routine tasks
    - Updating the time
    - Making sure the processor is not lost in a loop
  - Response to unexpected events
    - Power failure
    - Overheating

Interrupts

- Interrupt Terminology
  - Maskable vs. nonmaskable
    - Maskable – these interrupts can be selectively enabled or disabled
    - Nonmaskable – these cannot be disabled
  - Pending
    - An interrupt that has occurred but has not yet been serviced by the processor
    - Several interrupts can be pending at any given time
      - various rules to determine the priority

Interrupts

- Interrupts
  - When a specific condition is true
    - The processor or peripheral sets a FLAG
  - Two methods to respond to a FLAG
    - Polling
    - Interrupt Servicing

Interrupts

- Interrupts - Polling
  - The code must be written such that each interrupt flag is checked at the required frequency
  - When a flag is detected specific actions are taken
    - These actions are part of the program flow
  - When the required actions are complete the program continues

Interrupts

- Interrupts - Servicing
  - Each type of interrupt has its own reset vector
  - On an interrupt: PC ← Interrupt vector
  - Each interrupt has an enable bit
  - There is a global interrupt enable bit in the status register

Interrupts

Interrupt Process

Normal execution

Program Memory

Address (hex)	Instruction (hex)
0000 0000	0001 1000
0000 0004	0001 1010
0000 0008	0001 1020
0000 000C	0001 1040
0000 0010	0001 1060
…	
0000 1100	A23B 1254
0000 1102	23F4 C298
0000 1104	D246 7531
0000 1106	5679 0023
…	
0001 1018	0024 7654
0001 101C	9876 5432
0001 1020	AABC 4680
0001 1022	9854 2378
0001 1024	A765 567A
0001 1026	3490 6521
0001 1028	BB34 76CC
0001 102A	8732 2378

PC → 0000 1102

active instruction ← 0000 1100

Data Memory

(hex)	(hex)
2000 0000	2356 763A
2000 0004	23DC 43FA
2000 0008	98FC ACF4
…	
2000 1000	0000 2323
2000 1004	2222 5555
2000 1008	5454 3434
2000 100C	0909 0000
…	
2000 FFD8	X
2000 FFD4	X
2000 FFD8	X
2000 FFDC	X
…	
2000 FFF4	X
2000 FFF8	A345 B543
2000 FFFC	1245 6789

MSP → 2000 FFF8

Interrupts

Interrupt Process

Program Memory

Address (hex)	Instruction (hex)
0000 0000	0001 1000
0000 0004	0001 1010
0000 0008	0001 1020
0000 000C	0001 1040
0000 0010	0001 1060
…	
0000 1100	A23B 1254
0000 1102	23F4 C298
0000 1104	D246 7531
0000 1106	5679 0023
…	
0001 1018	0024 7654
0001 101C	9876 5432
0001 1020	AABC 4680
0001 1022	9854 2378
0001 1024	A765 567A
0001 1026	3490 6521
0001 1028	BB34 76CC
0001 102A	8732 2378

PC → 0000 1104

active instruction ← 0000 1102

Normal execution

Data Memory

(hex)	(hex)
2000 0000	2356 763A
2000 0004	23DC 43FA
2000 0008	98FC ACF4
…	
2000 1000	0000 2323
2000 1004	2222 5555
2000 1008	5454 3434
2000 100C	0909 0000
…	
2000 FFD8	X
2000 FFD4	X
2000 FFD8	X
2000 FFDC	X
…	
2000 FFF4	X
2000 FFF8	A345 B543
2000 FFFC	1245 6789

MSP → 2000 FFF8

Interrupts

Interrupt Process

Program Memory

Address (hex)	Instruction (hex)
0000 0000	0001 1000
0000 0004	0001 1010
0000 0008	0001 1020
0000 000C	0001 1040
0000 0010	0001 1060
…	
0000 1100	A23B 1254
0000 1102	23F4 C298
0000 1104	D246 7531
0000 1106	5679 0023
…	
0001 1018	0024 7654
0001 101C	9876 5432
0001 1020	AABC 4680
0001 1022	9854 2378
0001 1024	A765 567A
0001 1026	3490 6521
0001 1028	BB34 76CC
0001 102A	8732 2378

PC → 0000 1104

active instruction ← 0000 1102

Beginning of Interrupt

Flag set

Interrupt → pending

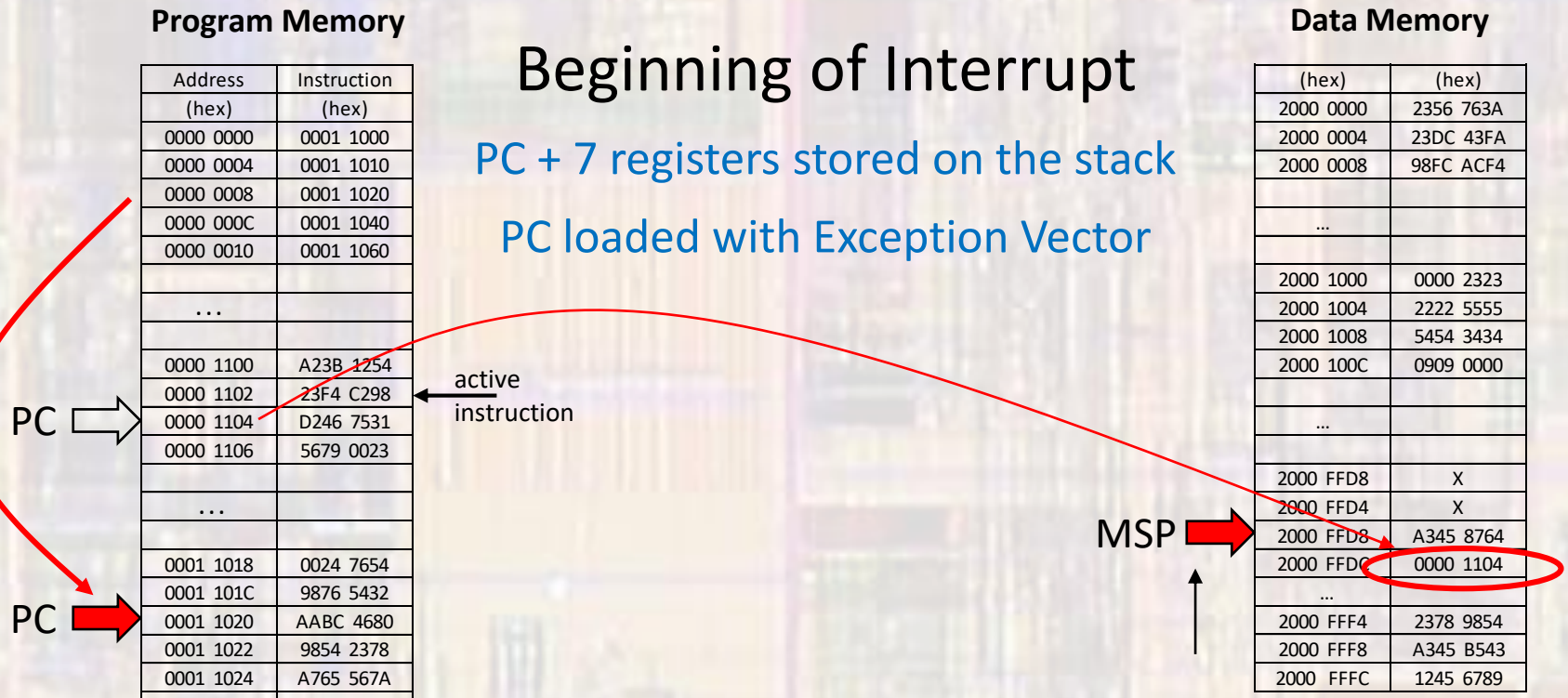
Data Memory

(hex)	(hex)
2000 0000	2356 763A
2000 0004	23DC 43FA
2000 0008	98FC ACF4
…	
2000 1000	0000 2323
2000 1004	2222 5555
2000 1008	5454 3434
2000 100C	0909 0000
…	
2000 FFD8	X
2000 FFD4	X
2000 FFD8	X
2000 FFDC	X
…	
2000 FFF4	X
2000 FFF8	A345 B543
2000 FFFC	1245 6789

MSP → 2000 FFF8

Interrupts

Interrupt Process



Interrupts

Interrupt Process

Program Memory

Address (hex)	Instruction (hex)
0000 0000	0001 1000
0000 0004	0001 1010
0000 0008	0001 1020
0000 000C	0001 1040
0000 0010	0001 1060
…	
0000 1100	A23B 1254
0000 1102	23F4 C298
0000 1104	D246 7531
0000 1106	5679 0023
…	
0001 1018	0024 7654
0001 101C	9876 5432
0001 1020	AABC 4680
0001 1022	9854 2378
0001 1024	A765 567A
0001 1026	3490 6521
0001 1028	BB34 76CC
0001 102A	8732 2378

PC

active instruction

Interrupt Service Routine (ISR)

Data Memory

(hex)	(hex)
2000 0000	2356 763A
2000 0004	23DC 43FA
2000 0008	98FC ACF4
…	
2000 1000	0000 2323
2000 1004	2222 5555
2000 1008	5454 3434
2000 100C	0909 0000
…	
2000 FFD8	X
2000 FFD4	X
2000 FFD8	A345 8764
2000 FFDC	0000 1104
…	
2000 FFF4	2378 9854
2000 FFF8	A345 B543
2000 FFFC	1245 6789

MSP

Interrupts

Interrupt Process

Program Memory

Address (hex)	Instruction (hex)
0000 0000	0001 1000
0000 0004	0001 1010
0000 0008	0001 1020
0000 000C	0001 1040
0000 0010	0001 1060
…	
0000 1100	A23B 1254
0000 1102	23F4 C298
0000 1104	D246 7531
0000 1106	5679 0023
…	
0001 1018	0024 7654
0001 101C	9876 5432
0001 1020	AABC 4680
0001 1022	9854 2378
0001 1024	A765 567A
0001 1026	3490 6521
0001 1028	BB34 76CC
0001 102A	8732 2378

PC

active instruction

Interrupt Service Routine (ISR)

Data Memory

(hex)	(hex)
2000 0000	2356 763A
2000 0004	23DC 43FA
2000 0008	98FC ACF4
…	
2000 1000	0000 2323
2000 1004	2222 5555
2000 1008	5454 3434
2000 100C	0909 0000
…	
2000 FFD8	X
2000 FFD4	X
2000 FFD8	A345 8764
2000 FFDC	0000 1104
…	
2000 FFF4	2378 9854
2000 FFF8	A345 B543
2000 FFFC	1245 6789

MSP

Interrupts

Interrupt Process

Program Memory

Address (hex)	Instruction (hex)
0000 0000	0001 1000
0000 0004	0001 1010
0000 0008	0001 1020
0000 000C	0001 1040
0000 0010	0001 1060
…	
0000 1100	A23B 1254
0000 1102	23F4 C298
0000 1104	D246 7531
0000 1106	5679 0023
…	
0001 1018	0024 7654
0001 101C	9876 5432
0001 1020	AABC 4680
0001 1022	9854 2378
0001 1024	A765 567A
0001 1026	3490 6521
0001 1028	BB34 76CC
0001 102A	8732 2378

PC

active instruction

Return from Interrupt

(BX LR)

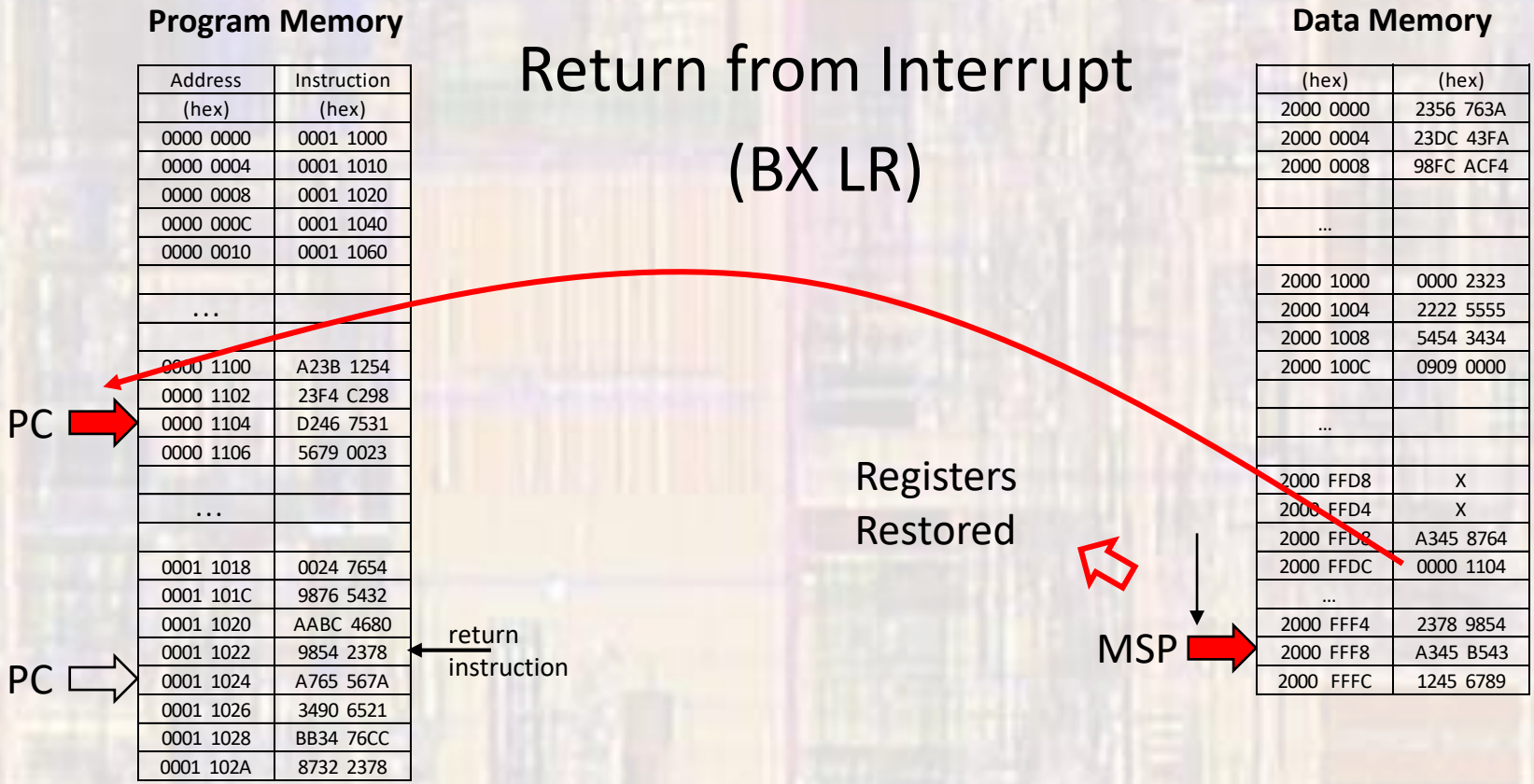
Data Memory

(hex)	(hex)
2000 0000	2356 763A
2000 0004	23DC 43FA
2000 0008	98FC ACF4
…	
2000 1000	0000 2323
2000 1004	2222 5555
2000 1008	5454 3434
2000 100C	0909 0000
…	
2000 FFD8	X
2000 FFD4	X
2000 FFD8	A345 8764
2000 FFDC	0000 1104
…	
2000 FFF4	2378 9854
2000 FFF8	A345 B543
2000 FFFC	1245 6789

MSP

Interrupts

Interrupt Process



Interrupts

Interrupt Process

Normal Execution

Program Memory

Address (hex)	Instruction (hex)
0000 0000	0001 1000
0000 0004	0001 1010
0000 0008	0001 1020
0000 000C	0001 1040
0000 0010	0001 1060
…	
0000 1100	A23B 1254
0000 1102	23F4 C298
0000 1104	D246 7531
0000 1106	5679 0023
…	
0001 1018	0024 7654
0001 101C	9876 5432
0001 1020	AABC 4680
0001 1022	9854 2378
0001 1024	A765 567A
0001 1026	3490 6521
0001 1028	BB34 76CC
0001 102A	8732 2378

PC → 0000 1106

active instruction ← 0000 1104

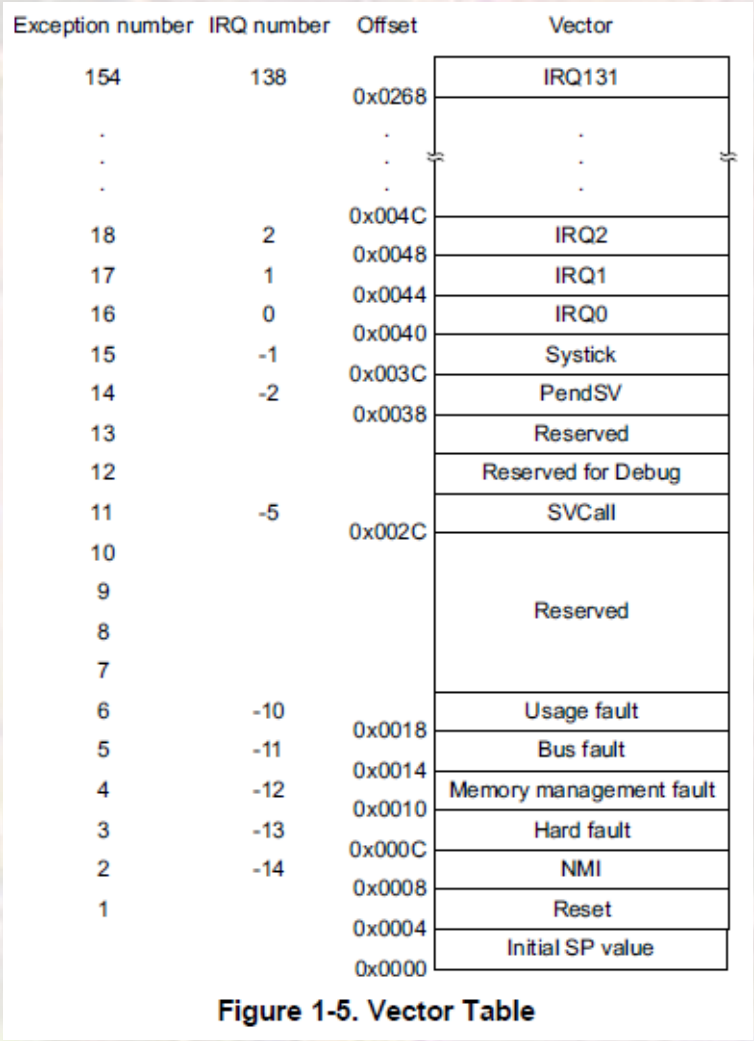
Data Memory

(hex)	(hex)
2000 0000	2356 763A
2000 0004	23DC 43FA
2000 0008	98FC ACF4
…	
2000 1000	0000 2323
2000 1004	2222 5555
2000 1008	5454 3434
2000 100C	0909 0000
…	
2000 FFD8	X
2000 FFD4	X
2000 FFD8	A345 8764
2000 FFDC	0000 1104
…	
2000 FFF4	2378 9854
2000 FFF8	A345 B543
2000 FFFC	1245 6789

MSP → 2000 FFF4

Interrupts

- Interrupt Vectors Table



Exception number	IRQ number	Offset	Vector
154	138	0x0268	IRQ131
.		.	.
.		.	.
.		.	.
		0x004C	
18	2	0x0048	IRQ2
17	1	0x0044	IRQ1
16	0	0x0040	IRQ0
15	-1	0x003C	Systick
14	-2	0x0038	PendSV
13			Reserved
12			Reserved for Debug
11	-5	0x002C	SVCall
10			
9			Reserved
8			
7			
6	-10	0x0018	Usage fault
5	-11	0x0014	Bus fault
4	-12	0x0010	Memory management fault
3	-13	0x000C	Hard fault
2	-14	0x0008	NMI
1		0x0004	Reset
		0x0000	Initial SP value

Figure 1-5. Vector Table

Note: Exception # and IRQ # do not match

System level events -- faults --

Interrupts

- Interrupt Vectors

NVIC INTERRUPT INPUT	SOURCE	FLAGS IN SOURCE
INTISR[0]	PSS (1)	
INTISR[1]	CS (1)	
INTISR[2]	PCM (1)	
INTISR[3]	WDT_A	
INTISR[4]	FPU_INT (2)	Combined interrupt from flags in the FPSCR (part of Cortex-M4 FPU)
INTISR[5]	FLCTL	Flash Controller interrupt flags
INTISR[6]	COMP_E0	Comparator_E0 interrupt flags
INTISR[7]	COMP_E1	Comparator_E1 interrupt flags
INTISR[8]	Timer_A0	TA0CCTL0.CCIFG
INTISR[9]	Timer_A0	TA0CCTLx.CCIFG (x = 1 through 4), TA0CTL.TAIFG
INTISR[10]	Timer_A1	TA1CCTL0.CCIFG
INTISR[11]	Timer_A1	TA1CCTLx.CCIFG (x = 1 through 4), TA1CTL.TAIFG
INTISR[12]	Timer_A2	TA2CCTL0.CCIFG
INTISR[13]	Timer_A2	TA2CCTLx.CCIFG (x = 1 through 4), TA2CTL.TAIFG
INTISR[14]	Timer_A3	TA3CCTL0.CCIFG
INTISR[15]	Timer_A3	TA3CCTLx.CCIFG (x = 1 through 4), TA3CTL.TAIFG
INTISR[16]	eUSCI_A0	UART or SPI mode TX, RX, and Status Flags
INTISR[17]	eUSCI_A1	UART or SPI mode TX, RX, and Status Flags
INTISR[18]	eUSCI_A2	UART or SPI mode TX, RX, and Status Flags
INTISR[19]	eUSCI_A3	UART or SPI mode TX, RX, and Status Flags
INTISR[20]	eUSCI_B0	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[21]	eUSCI_B1	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[22]	eUSCI_B2	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[23]	eUSCI_B3	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[24]	ADC14	IFG[0-31], LO/IN/HI-IFG, RDYIFG, OVIFG, TOVIFG

Interrupts

- Interrupt Vectors

NVIC INTERRUPT INPUT	SOURCE	FLAGS IN SOURCE
INTISR[25]	Timer32_INT1	Timer32 Interrupt for Timer1
INTISR[26]	Timer32_INT2	Timer32 Interrupt for Timer2
INTISR[27]	Timer32_INTC	Timer32 Combined Interrupt
INTISR[28]	AES256	AESRDYIFG
INTISR[29]	RTC_C	OFIFG, RDYIFG, TEVIFG, AIFG, RT0PSIFG, RT1PSIFG
INTISR[30]	DMA_ERR	DMA error interrupt
INTISR[31]	DMA_INT3	DMA completion interrupt3
INTISR[32]	DMA_INT2	DMA completion interrupt2
INTISR[33]	DMA_INT1	DMA completion interrupt1
INTISR[34]	DMA_INT0(3)	DMA completion interrupt0
INTISR[35]	I/O Port P1	P1IFG.x (x = 0 through 7)
INTISR[36]	I/O Port P2	P2IFG.x (x = 0 through 7)
INTISR[37]	I/O Port P3	P3IFG.x (x = 0 through 7)
INTISR[38]	I/O Port P4	P4IFG.x (x = 0 through 7)
INTISR[39]	I/O Port P5	P5IFG.x (x = 0 through 7)
INTISR[40]	I/O Port P6	P6IFG.x (x = 0 through 7)
INTISR[41]	Reserved	

Interrupts

- Interrupt Vector Handlers

```
void (* const interruptVectors[])(void) =
{
   (void (*)(void))((uint32_t)&__STACK_END),
                /* The initial stack pointer */
   Reset_Handler,              /* The reset handler       */
   NMI_Handler,                /* The NMI handler         */
   HardFault_Handler,          /* The hard fault handler  */
   MemManage_Handler,          /* The MPU fault handler   */
   BusFault_Handler,           /* The bus fault handler   */
   UsageFault_Handler,         /* The usage fault handler */
   0,                          /* Reserved                */
   0,                          /* Reserved                */
   0,                          /* Reserved                */
   0,                          /* Reserved                */
   SVC_Handler,                /* SVCall handler          */
   DebugMon_Handler,           /* Debug monitor handler   */
   0,                          /* Reserved                */
   PendSV_Handler,             /* The PendSV handler      */
   SysTick_Handler,            /* The SysTick handler     */
   PSS_IRQHandler,             /* PSS ISR                 */
   CS_IRQHandler,              /* CS ISR                  */
   PCM_IRQHandler,             /* PCM ISR                 */
   WDT_A_IRQHandler,           /* WDT ISR                 */
   FPU_IRQHandler,             /* FPU ISR                 */
   FLCTL_IRQHandler,           /* FLCTL ISR               */
   COMP_E0_IRQHandler,         /* COMP0 ISR               */
   COMP_E1_IRQHandler,         /* COMP1 ISR               */
   TA0_0_IRQHandler,           /* TA0_0 ISR               */
   TA0_N_IRQHandler,           /* TA0_N ISR               */
   TA1_0_IRQHandler,           /* TA1_0 ISR               */
   TA1_N_IRQHandler,           /* TA1_N ISR               */
   TA2_0_IRQHandler,           /* TA2_0 ISR               */
   TA2_N_IRQHandler,           /* TA2_N ISR               */
   TA3_0_IRQHandler,           /* TA3_0 ISR               */
   TA3_N_IRQHandler,           /* TA3_N ISR               */
   EUSCIA0_IRQHandler,         /* EUSCIA0 ISR             */
   EUSCIA1_IRQHandler,         /* EUSCIA1 ISR             */
   EUSCIA2_IRQHandler,         /* EUSCIA2 ISR             */
   EUSCIA3_IRQHandler,         /* EUSCIA3 ISR             */
   EUSCIB0_IRQHandler,         /* EUSCIB0 ISR             */
   EUSCIB1_IRQHandler,         /* EUSCIB1 ISR             */
   EUSCIB2_IRQHandler,         /* EUSCIB2 ISR             */
   EUSCIB3_IRQHandler,         /* EUSCIB3 ISR             */
   ADC14_IRQHandler,           /* ADC14 ISR               */
   T32_INT1_IRQHandler,        /* T32_INT1 ISR            */
   T32_INT2_IRQHandler,        /* T32_INT2 ISR            */
   T32_INTC_IRQHandler,        /* T32_INTC ISR            */
   AES256_IRQHandler,          /* AES ISR                 */
   RTC_C_IRQHandler,           /* RTC ISR                 */
   DMA_ERR_IRQHandler,         /* DMA_ERR ISR             */
   DMA_INT3_IRQHandler,        /* DMA_INT3 ISR            */
   DMA_INT2_IRQHandler,        /* DMA_INT2 ISR            */
   DMA_INT1_IRQHandler,        /* DMA_INT1 ISR            */
   DMA_INT0_IRQHandler,        /* DMA_INT0 ISR            */
   PORT1_IRQHandler,           /* PORT1 ISR               */
   PORT2_IRQHandler,           /* PORT2 ISR               */
   PORT3_IRQHandler,           /* PORT3 ISR               */
   PORT4_IRQHandler,           /* PORT4 ISR               */
   PORT5_IRQHandler,           /* PORT5 ISR               */
   PORT6_IRQHandler            /* PORT6 ISR               */
};
```

Interrupts

- Interrupt Operation
  - 4 steps
    - Enable interrupts in the NVIC
    - Set priorities for each interrupt source
      - only upper nibble used
    - Enable individual peripheral interrupt
    - Write an Interrupt Service Routine
      - Be sure to clear the interrupt flag
  - Make sure global interrupts are enabled

Interrupts

- External Interrupts
  - 64 pins can be used as external interrupts
  - Port1.x → Port6.x
  - Rising or Falling edges
  - Lower pin numbers in a port have higher priority
    - P6.0 has higher priority than P6.3
  - Key Registers
    - PxIE Interrupt enable
    - PxIES Rising/falling
    - PxIFG Interrupt Flag
    - PxIV Interrupt Source Vector (pin 0-7)

NVIC

- Nested Vector Interrupt Controller – NVIC
  - 64 interrupts.
  - Programmable priority level of 0-7 for each interrupt
    - A higher level corresponds to a lower priority
    - Level 0 is the highest interrupt priority
  - Low-latency exception and interrupt handling.
  - Level and pulse detection of interrupt signals.
  - Dynamic re-prioritization of interrupts.
  - Grouping of priority values into group priority and subpriority fields.
  - Interrupt tail-chaining.
  - An external Non-maskable interrupt (NMI)

NVIC

- Nested Vector Interrupt Controller – NVIC
  - Interrupts are enabled / disabled on a peripheral basis
  - Some interrupts are not maskable (disable-able)
  - Enabled interrupts are either:
    - inactive – not currently active
    - pending – triggered but not yet processed
    - active – currently being processed in an interrupt service routine (ISR)
  - Level sensitive interrupts must be explicitly reset
  - Edge triggered interrupts are cleared at the beginning of the ISR
    - This means a new trigger event could occur during the ISR – restarting the ISR immediately upon completion of the current ISR

NVIC

- NVIC – Non-maskable interrupts
  - External NMI pin (if configured in NMI mode)
  - Oscillator fault condition
  - Power Supply System (PSS) generated interrupts
  - Power Control Manager (PCM) generated interrupts

NVIC

- NVIC –User interrupts (Maskable)
  - PSS
  - CS
  - PCM
  - WDT
  - FPU
  - FLCTL
  - COMP
  - Timer_A
  - eUSCI
  - ADC14
  - Timer32
  - AES
  - RTC
  - DMA
  - I/O Ports

NVIC

- NVIC - Registers

actual structure name

Table 2-25. NVIC Registers

Offset	Acronym	Register Name	Type	Reset	Section
100h	ISER0	Irq 0 to 31 Set Enable Register	read-write	00000000h	Section 2.4.3.1
104h	ISER1	Irq 32 to 63 Set Enable Register	read-write	00000000h	Section 2.4.3.2
180h	ICER0	Irq 0 to 31 Clear Enable Register	read-write	00000000h	Section 2.4.3.3
184h	ICER1	Irq 32 to 63 Clear Enable Register	read-write	00000000h	Section 2.4.3.4
200h	ISPR0	Irq 0 to 31 Set Pending Register	read-write	00000000h	Section 2.4.3.5
204h	ISPR1	Irq 32 to 63 Set Pending Register	read-write	00000000h	Section 2.4.3.6
280h	ICPR0	Irq 0 to 31 Clear Pending Register	read-write	00000000h	Section 2.4.3.7
284h	ICPR1	Irq 32 to 63 Clear Pending Register	read-write	00000000h	Section 2.4.3.8
300h	IABR0	Irq 0 to 31 Active Bit Register	read-only	00000000h	Section 2.4.3.9
304h	IABR1	Irq 32 to 63 Active Bit Register	read-only	00000000h	Section 2.4.3.10
400h	IPR0	Irq 0 to 3 Priority Register	read-write	00000000h	Section 2.4.3.11
404h	IPR1	Irq 4 to 7 Priority Register	read-write	00000000h	Section 2.4.3.12
408h	IPR2	Irq 8 to 11 Priority Register	read-write	00000000h	Section 2.4.3.13
40Ch	IPR3	Irq 12 to 15 Priority Register	read-write	00000000h	Section 2.4.3.14
410h	IPR4	Irq 16 to 19 Priority Register	read-write	00000000h	Section 2.4.3.15
414h	IPR5	Irq 20 to 23 Priority Register			
418h	IPR6	Irq 24 to 27 Priority Register			
41Ch	IPR7	Irq 28 to 31 Priority Register			
420h	IPR8	Irq 32 to 35 Priority Register			
424h	IPR9	Irq 36 to 39 Priority Register			
428h	IPR10	Irq 40 to 43 Priority Register			
42Ch	IPR11	Irq 44 to 47 Priority Register			
430h	IPR12	Irq 48 to 51 Priority Register	read-write	00000000h	Section 2.4.3.23
434h	IPR13	Irq 52 to 55 Priority Register	read-write	00000000h	Section 2.4.3.24
438h	IPR14	Irq 56 to 59 Priority Register	read-write	00000000h	Section 2.4.3.25
43Ch	IPR15	Irq 60 to 63 Priority Register	read-write	00000000h	Section 2.4.3.26
F00h	STIR	Software Trigger Interrupt Register	write-only	00000000h	Section 2.4.3.27

Note – even though the 8b IP registers are packed 4 to a 32b register they are accessed independently in the C structure

NVIC->ISER[0]

NVIC->ICER[0]

NVIC->ISPR[0]

NVIC->ICPR[0]

NVIC->IABR[0]

NVIC->IP[0]

NVIC->IP[63]

NVIC

- NVIC – Interrupt Set Enable Registers

Figure 2-20. ISER0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA																															
R/W-0h																															

Table 2-26. ISER0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SETENA	R/W	0h	Writing 0 to a SETENA bit has no effect, writing 1 to a bit enables the corresponding interrupt. Reading the bit returns its current enable state. Reset clears the SETENA fields.

Figure 2-21. ISER1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA																															
R/W-0h																															

Table 2-27. ISER1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SETENA	R/W	0h	Writing 0 to a SETENA bit has no effect, writing 1 to a bit enables the corresponding interrupt. Reading the bit returns its current enable state. Reset clears the SETENA fields.

NVIC

- NVIC – Interrupt Clear Enable Registers

Figure 2-22. ICER0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRENA																															
R/W-0h																															

Table 2-28. ICER0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLRENA	R/W	0h	Writing 0 to a CLRENA bit has no effect, writing 1 to a bit disables the corresponding interrupt. Reading the bit returns its current enable state. Reset clears the CLRENA field.

Figure 2-23. ICER1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRENA																															
R/W-0h																															

Table 2-29. ICER1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLRENA	R/W	0h	Writing 0 to a CLRENA bit has no effect, writing 1 to a bit disables the corresponding interrupt. Reading the bit returns its current enable state. Reset clears the CLRENA field.

NVIC

- NVIC – Interrupt Set Pending Registers

Figure 2-24. ISPR0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETPEND																															
R/W-0h																															

Table 2-30. ISPR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SETPEND	R/W	0h	Writing 0 to a SETPEND bit has no effect, writing 1 to a bit pends the corresponding interrupt. Reading the bit returns its current state.

Figure 2-25. ISPR1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETPEND																															
R/W-0h																															

Table 2-31. ISPR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SETPEND	R/W	0h	Writing 0 to a SETPEND bit has no effect, writing 1 to a bit pends the corresponding interrupt. Reading the bit returns its current state.

NVIC

- NVIC – Interrupt Clear Pending Registers

Figure 2-26. ICPR0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRPEND																															
R/W-0h																															

Table 2-32. ICPR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLRPEND	R/W	0h	Writing 0 to a CLRPEND bit has no effect, writing 1 to a bit clears the corresponding pending interrupt. Reading the bit returns its current state.

Figure 2-27. ICPR1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CLRPEND																															
R/W-0h																															

Table 2-33. ICPR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	CLRPEND	R/W	0h	Writing 0 to a CLRPEND bit has no effect, writing 1 to a bit clears the corresponding pending interrupt. Reading the bit returns its current state.

NVIC

- NVIC – Interrupt Active Bit Registers

Figure 2-28. IABR0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTIVE																															
R-0h																															

Table 2-34. IABR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ACTIVE	R	0h	Interrupt active flags. Reading 0 implies the interrupt is not active or stacked. Reading 1 implies the interrupt is active or pre-empted and stacked.

Figure 2-29. IABR1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACTIVE																															
R-0h																															

Table 2-35. IABR1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	ACTIVE	R	0h	Interrupt active flags. Reading 0 implies the interrupt is not active or stacked. Reading 1 implies the interrupt is active or pre-empted and stacked.

NVIC

- NVIC – Interrupt Priority Registers (0 -15)

Figure 2-30. IPR0 Register

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
PRI_3	PRI_2	PRI_1	PRI_0
R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 2-36. IPR0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_3	R/W	0h	Priority of interrupt 3
23-16	PRI_2	R/W	0h	Priority of interrupt 2
15-8	PRI_1	R/W	0h	Priority of interrupt 1
7-0	PRI_0	R/W	0h	Priority of interrupt 0

Note:
The structure uses 8 bit values labeled IP[0] – IP[63)

To set int 2 priority NVIC->IP[2]

• • •

Figure 2-45. IPR15 Register

31 30 29 28 27 26 25 24	23 22 21 20 19 18 17 16	15 14 13 12 11 10 9 8	7 6 5 4 3 2 1 0
PRI_63	PRI_62	PRI_61	PRI_60
R/W-0h	R/W-0h	R/W-0h	R/W-0h

Table 2-51. IPR15 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-24	PRI_63	R/W	0h	Priority of interrupt 63
23-16	PRI_62	R/W	0h	Priority of interrupt 62
15-8	PRI_61	R/W	0h	Priority of interrupt 61
7-0	PRI_60	R/W	0h	Priority of interrupt 60

NVIC

- NVIC – Software Trigger Interrupt Register

Figure 2-46. STIR Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESERVED																							INTID								
W-0h																							W-0h								

Table 2-52. STIR Register Field Descriptions

Bit	Field	Type	Reset	Description
31-9	RESERVED	W	0h	
8-0	INTID	W	0h	Interrupt ID field. Writing a value to the INTID field is the same as manually pending an interrupt by setting the corresponding interrupt bit in an Interrupt Set Pending Register.

NVIC

- NVIC – Software Trigger Interrupt Register

Table 6-39. NVIC Interrupts

NVIC INTERRUPT INPUT	SOURCE	FLAGS IN SOURCE
INTISR[0]	PSS (1)	
INTISR[1]	CS (1)	
INTISR[2] INTISR[3]	PCM (1) WDT_A	
INTISR[4]	FPU_INT (2)	Combined interrupt from flags in the FPSCR (part of Cortex-M4 FPU)
INTISR[5]	FLCTL	Flash Controller interrupt flags
INTISR[6]	COMP_E0	Comparator_E0 interrupt flags
INTISR[7]	COMP_E1	Comparator_E1 interrupt flags
INTISR[8]	Timer_A0	TA0CCTL0.CCIFG
INTISR[9]	Timer_A0	TA0CCTLx.CCIFG (x = 1 through 4), TA0CTL.TAIFG
INTISR[10]	Timer_A1	TA1CCTL0.CCIFG
INTISR[11]	Timer_A1	TA1CCTLx.CCIFG (x = 1 through 4), TA1CTL.TAIFG
INTISR[12]	Timer_A2	TA2CCTL0.CCIFG
INTISR[13]	Timer_A2	TA2CCTLx.CCIFG (x = 1 through 4), TA2CTL.TAIFG
INTISR[14]	Timer_A3	TA3CCTL0.CCIFG
INTISR[15]	Timer_A3	TA3CCTLx.CCIFG (x = 1 through 4), TA3CTL.TAIFG
INTISR[16]	eUSCI_A0	UART or SPI mode TX, RX, and Status Flags
INTISR[17]	eUSCI_A1	UART or SPI mode TX, RX, and Status Flags
INTISR[18]	eUSCI_A2	UART or SPI mode TX, RX, and Status Flags
INTISR[19]	eUSCI_A3	UART or SPI mode TX, RX, and Status Flags
INTISR[20]	eUSCI_B0	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[21]	eUSCI_B1	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[22]	eUSCI_B2	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)
INTISR[23]	eUSCI_B3	SPI or I^2C mode TX, RX, and Status Flags (I^2C in multiple-slave mode)

NVIC

- NVIC – Software Trigger Interrupt Register

Table 6-39. NVIC Interrupts

NVIC INTERRUPT INPUT	SOURCE	FLAGS IN SOURCE
INTISR[24]	ADC14	IFG[0-31], LO/IN/HI-IFG, RDYIFG, OVIFG, TOVIFG
INTISR[25]	Timer32_INT1	Timer32 Interrupt for Timer1
INTISR[26]	Timer32_INT2	Timer32 Interrupt for Timer2
INTISR[27]	Timer32_INTC	Timer32 Combined Interrupt
INTISR[28]	AES256	AESRDYIFG
INTISR[29]	RTC_C	OFIFG, RDYIFG, TEVIFG, AIFG, RT0PSIFG, RT1PSIFG
INTISR[30]	DMA_ERR	DMA error interrupt
INTISR[31]	DMA_INT3	DMA completion interrupt3
INTISR[32]	DMA_INT2	DMA completion interrupt2
INTISR[33]	DMA_INT1	DMA completion interrupt1
INTISR[34]	DMA_INT0(3)	DMA completion interrupt0
INTISR[35]	I/O Port P1	P1IFG.x (x = 0 through 7)
INTISR[36]	I/O Port P2	P2IFG.x (x = 0 through 7)
INTISR[37]	I/O Port P3	P3IFG.x (x = 0 through 7)
INTISR[38]	I/O Port P4	P4IFG.x (x = 0 through 7)
INTISR[39]	I/O Port P5	P5IFG.x (x = 0 through 7)
INTISR[40]	I/O Port P6	P6IFG.x (x = 0 through 7)
INTISR[41]	Reserved	
INTISR[42]	Reserved	

• • •

Interrupts

- Setup pin 23 to be an input with no pullup/down and configured as a rising edge interrupt, and use it to toggle pin 3

Pin Setup

```
void pin_setup(void){
  // setup pins
  // Note pin 23 is Port 6 Pin 1
  // Note pin 3 is Port 3 Pin 2
  P6->SEL0 &= ~0x02;      // standard pin functionality
  P6->SEL1 &= ~0x02;
  P6->DIR &= ~0x02;       // input
  P6->REN &= ~0x02;       // No pull
  P6->IES &= ~0x02;       // rising edge trigger
  P6->IFG &= ~0x02;       // Clear any accidental Flag
  P6->IE |= 0x02;         // enable interrupts

  P3->DIR |= 0x04;        // prepare LED output
  P3->OUT &= ~0x04;       // set starting point

  return;
}
```

- Px->SEL0, Px->SEL1 – Mode Select Register
- Px->IE – Interrupt Enable Register
  - "1" for enable
- Px->IES – Interrupt Edge Select Register
  - "0" for low → high, "1" for high → low
- Px->IFG – Interrupt Flag Register
  - Set to "1" on selected edge transition
- Px->IV – Interrupt Vector Register

Interrupts

- Setup pin 23 to be an input with no pullup/down and configured as a rising edge interrupt

NVIC Setup

INTISR[39]	I/O Port P5	P5IFG.x (x = 0 through 7)
INTISR[40]	I/O Port P6	P6IFG.x (x = 0 through 7)
INTISR[41]	Reserved	

reminder – pins are prioritized 0 – 7(H to L) in a port

Only upper nibble used

```
void NVIC_setup(void){
    // setup NVIC
    // Enable PORT6
    // Note: port 6 is INTISR(40)
    NVIC->IP[40] |= 0x20;              // Set a priority
    NVIC->ISER[1] |= 0x00000100;       // ISER1 starts at 32

    return;
}
```

43-42-41-40 39-38-37-36 35-34-33-32

Figure 2-20. ISER0 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA																															
R/W-0h																															

Table 2-26. ISER0 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SETENA	R/W	0h	Writing 0 to a SETENA bit has no effect, writing 1 to a bit enables the corresponding interrupt. Reading the bit returns its current enable state. Reset clears the SETENA fields.

Figure 2-21. ISER1 Register

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SETENA																															
R/W-0h																															

Table 2-27. ISER1 Register Field Descriptions

Bit	Field	Type	Reset	Description
31-0	SETENA	R/W	0h	Writing 0 to a SETENA bit has no effect, writing 1 to a bit enables the corresponding interrupt. Reading the bit returns its current enable state. Reset clears the SETENA fields.

Interrupts

- Setup pin 3 to be an output to toggle an LED

Interrupt Handler

```
// Setup the Interrupt Handler
void PORT6_IRQHandler(void){
    P3->OUT = ~P3->OUT;      // toggle output – LAZY Method
    P6->IFG &= ~0x02;        // clear the flag for pin 6.1

    return;
}
```

Interrupts

- Setup pin 23 to be an input with no pullup/down and configured as a rising edge interrupt

```
int main(void){
    // setup routines
    pin_setup();
    NVIC_setup();

    _enable_interrupts();

    while(1){
       ;    // do nothing in the main program
    } // end while

   return 0;
}
```

Main

Interrupts

- Put it all together

```
/*
 * interrupt_external_example.c
 *
 *  Created on: Jul 25, 2018
 *      Author: johnsontimoj
 */
//////////////////////////////////
//
// use pin 23 as an external edge interrupt
//
// toggle pin 3 in the ISR
//
//////////////////////////////////
// includes
#include "msp.h"
#include <stdio.h>

// prototypes
void pin_setup(void);
void NVIC_setup(void);

int main(void){
    // setup routines
    pin_setup();
    NVIC_setup();

    _enable_interrupts();

    while(1){
       ;    // do nothing in the main program
    } // end while

   return 0;
}
```

```
void pin_setup(void){
    // setup pins
    // Note pin 23 is Port 6 Pin 1
    // Note pin 3 is Port 3 Pin 2
    P6->SEL0 &= ~0x02;       // standard pin functionality
    P6->SEL1 &= ~0x02;
    P6->DIR &= ~0x02;        // input
    P6->REN &= ~0x02;        // No pull
    P6->IES &= ~0x02;        // rising edge trigger
    P6->IFG &= ~0x02;        // Clear any accidental Flag
    P6->IE |= 0x02;          // enable interrupts

    P3->DIR |= 0x04;         // prepare LED output
    P3->OUT &= ~0x04;        // set starting point

    return;
}

void NVIC_setup(void){
    // setup NVIC
    // Enable PORT6
    // Note: port 6 is INTISR(40)
    NVIC->IP[40] |= 0x20;            // Set a priority
    NVIC->ISER[1] |= 0x00000100;     // ISER1 starts at 32

    return;
}

// Setup the Interrupt Handler
void PORT6_IRQHandler(void){
    P3->OUT = ~P3->OUT;      // toggle output - LAZY Method
    P6->IFG &= ~0x02;        // clear the flag for pin 6.1

    return;
}
```

note: No prototype declared by msp.h