Embedded System HW 4

Zach Jackovich

1. What are the differences between GIE and TAIE? GIE is General Interrupt Enable and TAIE is Timer A Interrupt Enable, so the GIE is the universal interrupt enable for the MSP430 board and the Timer A interrupt enable is enabling interrupts for Timer A. The GIE must always be set to enable any interrupt.

2. What is ISR? Where are the ISRs located? Who finds the ISRs from its location? How are the ISRs found? ISR is the Interrupt Service Routine, which is the function that is performed when an interrupt is triggered.

3. Suppose we want to write a program that uses the interrupt generated by the TAIFG flag of Timer_A (module#0). Now answer the following questions:

a. How many interrupt enable bits do we need to enable for this purpose? What are those? We need to enable GIE but that is not a bit we need to set, you must use the intrinsic function "_enable_interrupts();" in your code to set the GIE. We do have a bit we need to set in the Timer A CTL register which is accessed using the TAIE bit mask, setting the 1st bit in TA0CTL.

b. Write the code to configure the TA0CTL register for this purpose (using ACLK, divided by 1, continuous mode, TAR cleared at the beginning).

```
TA0CTL |= (TASSEL_1 | ID_0 | MC_2 | TACLR);
```

4. The following table mentions the list of interrupt events associated with Timer_A0 of MSP430.

Event	Bits	Vector	MSP430 Policy
Rollback-to-zero	TAIE/TAIFG in TACTL	A1	Programmer clears the flag
Channel 1	CCIE/CCIFG in TACCTL1		(shared vector)
Channel 2	CCIE/CCIFG in TACCTL2		
Channel 0	CCIE/CCIFG in TACCTL0	A0	Hardware clears the flag (non-shared vector)

It shows, Timer_A0 has three channels (0, 1, 2), each of which can instigate an interrupt. Now, suppose TimerA_0 is configured to use ACLK, divided by 1, up mode (TA0CCR = 1024), TAR reset at beginning.

a. **If we want to use the interrupt caused by channel 0, what will be the value of TAR when the CCIFG is asserted? Which vector should the start address of ISR be linked to?**

   The value of TAR would be the same as TA0CCR, which in this case is 1024.

   The start address of the ISR should be linked to vector A0. The code associated with this vector would be the statement "**`#pragma vector = TIMER0_A0_VECTOR`**"

b. If we want to use the interrupt caused by channel 1, what will be the value of TAR when the CCIFG is asserted? Which vector should the start address of ISR be linked to?

The value of TAR would be the same as TA0CCR, which in this case is 1024.

The start address of the ISR should be linked to vector A0. The code associated with this vector would be the statement "**`#pragma vector = TIMER0_A1_VECTOR`**"

c. If we want to use the interrupt caused by channel 2, what will be the value of TAR when the CCIFG is asserted? Which vector should the start address of ISR be linked to?

The value of TAR would be the same as TA0CCR, which in this case is 1024.
The start address of the ISR should be linked to vector A0. The code associated with this vector would be the statement "**`#pragma vector = TIMER0_A1_VECTOR`**"

d. If we want to use the interrupt caused by the rollback-to-zero event, what will be the value of TAR when the TAIFG is asserted? Which vector should the start address of ISR be linked to?

The value of TAR would be the same as TA0CCR, which in this case is 1024.
The start address of the ISR should be linked to vector A0. The code associated with this vector would be the statement "**`#pragma vector = TIMER0_A1_VECTOR`**"

Mode	MCLK	SMCLK	ACLK
Active	On	On	On
LPM0	x	On	On
LPM3	x	x	On
LPM4	x	x	x

5. The table above shows the low power modes in MSP430.

a. Which mode, if activated, will consume the maximum power? Why?

Active mode will consume the most power, because all 3 clock sources will be running.

b. Which mode, if activated, will consume the minimum power? Why?

Low Power Mode 4 (LPM4) will consume the least power, because all clock sources will be shut off. This is considered the "Deep sleep" mode, where the CPU is basically completely asleep.

6. How many bits in the status register (SR) control the low power modes? What are they? Do we need to use the associated SR bits to enable or disable the low power modes? Why?

4 bits, bits 4, 5, 6, and 7 control the low power modes in the Status Register (SR). We DO NOT use these bits to enable or disable the low power modes, because this status register is inside the CPU and the registers inside the CPU do not have addresses. Because they do not have addresses, we need to use intrinsic functions built into C to turn low power modes on. We use the function call, "**`_low_power_modes_x();`**" where the x refers to which low power mode we want to use, to turn this low power mode on in our code.

7. How does the CPU wake up from low power modes?

Interrupts are the only way to "wake up" the CPU/MCLK when low power modes are active.