Mark Anthony Serrano (mmserran@ucsc.edu)
Lab Section: 2 MW TA: Jay Roldan
Due 12/12/2012
Lab Partner: None

Title:

Lab 6: Timers and Interrupts

Purpose:

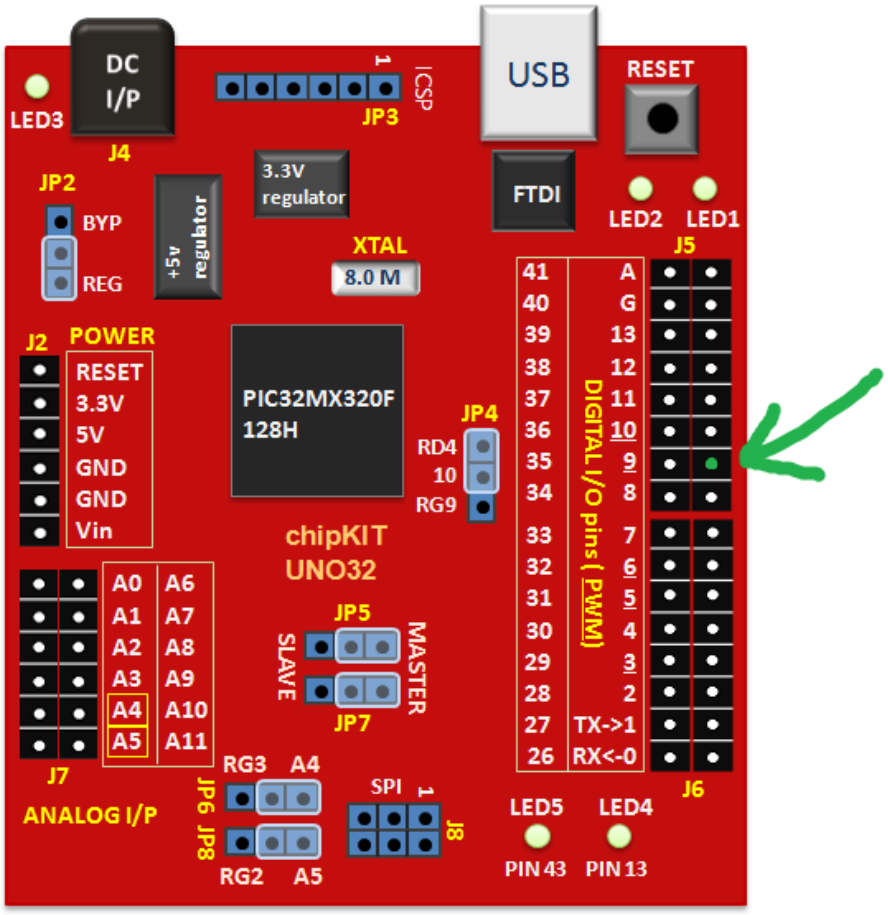
The purpose of this lab is to gain an intimate understanding of timers, interrupts, and communicating with the hardware through software.

Procedure:

In order to reproduce the results of this lab, understanding the I/O pins is crucial. It is required by lab 6 to utilize pin 9 of the chipKIT Uno32 board. Taken from reference [1], Figure 1 shows where pin 9 is located on the chipKIT Uno32 board. Then utilizing the "Pinout Table by Logical Pin Number" located on page 12 of the chipKIT Uno32 Board Reference Manual [2], we see that the chipKIT pin 9 corresponds to the PIC32 pin 51. Furthermore utilizing table 1-1 from Microchip PIC32MXX3XXX/4XX Data Sheet [3], we see that the PIC32 pin corresponds to RD3 of Port D. Using the correct bit masks and addresses, TRISD3 was cleared to set chipKIT pin 9 as an output; completing the SetupPort subroutine. To complete the SetupTimer subroutine, Timer 1 was setup as seen in Figure 2 while saving steps 5 and 7 for use in our PlayNote subroutine. The pseudo-code for the PlayNote subroutine can be seen in Figure 3. The final subroutine, the interrupt service routine for Timer 1 (T1_ISR), follows the register saving convention, inverts the output of chipKIT pin 9 (PIC32 PORTD, RD3), and then re-enables interrupts. Two helper functions, EnableTimer and DisableTimer, were created to shorten the code. Both turn on/off Timer 1 by setting or clearing bit<15> of T1CON. DisableTimer, however, also clears the TM1 count to follow the convention of cleaning up after using something. The extra credit melody, was created by translating piano tabs (Figure 4) to the main.cpp representation of a tune, but with a full note of size 200 (#define FULL_NOTE 200).

Algorithms and other data:

Figure 1:



layout of major components, courtesy of reference [1]

Figure 2:

14.3.4.2 16-BIT SYNCHRONOUS COUNTER INITIALIZATION STEPS

The following steps need to be performed to configure the timer for 16-bit Synchronous Timer mode.

1. Clear the ON control bit (TxCON<15> = 0) to disable the timer.
2. Clear the TCS control bit (TxCON<1> = 0) to select the internal PBCLK source.
3. Select the desired timer input clock prescale.
4. Load/Clear the timer register TMRx.
5. Load the period register PRx with the desired 16-bit match value.
6. If interrupts are used:
   a) Clear the TxIF interrupt flag bit in the IFSx register.
   b) Configure the interrupt priority and subpriority levels in the IPCx register.
   c) Set the TxIE interrupt enable bit in the IECx register.
7. Set the ON control bit (TxCON<15> = 1) to enable the timer.

Figure 3:

```
/* store registers to stack */

/*** Calculate Tone Period ************************************/
/* original formula: Period = (80,000,000/256)*(1/2)*(1/freq)  */
/* implemented as:          = (40,000,000/256*freq) */

    make divisor: $s0 = 256*frequency
    divide dividend by divisor: $s0 = 40,000,000/$s0
    Set PR1 - set the period
/*************************************************************/

/*** Tone Duration ********************************************/
    enable Timer1
    tone duration delay using Timer0
    disable Timer1
/*************************************************************/

/*** Silence Duration *****************************************/
    calculate the amount of silence after tone
    use delay subroutine from Timer0
/*************************************************************/

/* restore registers to stack */
```

Figure 4:

```
4|ee-e-ce-g---------||c----------------e-ga-fg-e-cd----||
3|-------------g----||---g--e--a-b-Aa-g-----------b---||
2|-------------g----||--------------------------------||
                    |-------------Repeat-------------|


4|-gFfD-e--c---cde|-gFfD-e--c-cc----|-gFfD-e--c---cd-e|-D--d--c--------|
3|--------a--a----|---------c-cc----|--------a--a-----|---------------|
2|----------------|-----------------|-----------------|---------------|



4|-cc-c-cd-ec------|-cc-c-cd-e------|-cc-c-cd-ec------|-ee-e-ce-g-------|
3|------------ag---|---------------|------------ag---|------------g----|
2|-----------------|---------------|-----------------|-----------------|


4|-ec-------fff---|---aaa-gfe------|-ec-------fff---|----fff-edc-----|
3|----gg-a--------|-b-------------|----gg-a--------|-b-------------|
2|----------------|---------------|----------------|----------------|
```

The "Super Mario Bros" Theme Song, courtesy of reference [4].

What went wrong or what were the challenges:

The biggest challenge was knowing how to interface with the hardware. For the majority of this lab, I did not know which pin was what. It wasn't until I figured out that the Uno32 J5-3 connector is the chipKIT pin 9 which connected to the PIC32 pin 51, which corresponds RD3 on PORTD. This allowed me to wire the board correctly and use the appropriate TRISx address.

Other information:

How did you go about designing your program?

- To design my program, I utilized the manuals available (such as Figure 2) and made sure each step was executed at some point in the correct subroutine.

How did you convert the frequency to a note duration?

- To convert a frequency to a note duration, we let timer 1 drive the frequency tone and let timer 0 control the note duration. Hence, the frequency was never converted but rather produced by timer 1 while timer 0 controlled the duration.

How did you implement the silent part of a note after a tone?

- PlayNote accepts 3 arguments, in particular $a2 holds the full note duration while $a1 holds the tone duration. By delaying ($a2-$a1) milliseconds, we will have complete the duration of a full note.

What sort of bugs did you encounter while writing your program?

- The main bugs of my program were using the wrong addresses.

Using the binutils assembler (as) manual, what are the xaw arguments to the .section statement for the vector address?

```
16  .section .vector_4,"xaw"
17      j T1_ISR
18      nop
```

- On line 16, the optional argument "xaw" represents a series of flags. Here, "x" means executable section, "a" means ignored (For compatibility with the ELF version), and "w" means writable section.

Why do we need the extra 1/2 when calculating the prescalar register value?

- Using the factor (1/2) when calculating the PR1 value accounts for the fact that we need to switch chipKIT pin 9 on and off. This is because one period includes both turning pin 9 on, and turning pin 9 off.

Conclusion:

This was probably my favorite lab. Learning how to use a microprocessor as well as peripherals for it was very significant for me. Now that I am knowledgeable with microprocessors and how to read their manuals, I'm excited to see how I can expand my chipKIT Uno32 board. And although I had struggled with this far longer than I anticipated, it also taught me about how to properly isolate a problem when debugging.

References:

[1] http://embedded-lab.com/blog/?p=4444

[2] http://www.digilentinc.com/Data/Products/CHIPKIT-UNO32/chipKITUno32_rm.pdf

[3] http://ww1.microchip.com/downloads/en/DeviceDoc/61143H.pdf

[4] http://tabnabber.com/