Benjamin Linam

Bml42@students.uwf.edu

EEL 4744L: Microprocessor Applications Laboratory

Lab 9: Input Capture

4/23/2018

Objective

The object of this lab is to introduce students to the input capture function of the 68HC11.

Introduction/Background/Theory

The 68HC11 has three input capture functions that interpret input signal transitions such as the rising or falling edges of a wave. The HC11 uses an internal time counter (TCNT) of which time stamps can be recorded and saved for later comparisons. This lab requires use of input capture 1 (IC1) to receive a periodic signal and determine its period and frequency by utilizing the interrupt method where an interrupt will be triggered for each rising edge of the signal for recording time stamps.

Procedure

1. Since the program uses the interrupt method, the interrupt vector is initialized, followed by the stack and initialization of IC1 in the IC1_init subroutine.

2. Figure 1 shows that the program enables interrupts, waits for one second and then tests to see if an input has been received and two rising edges have been captured.

```
                        ORG     $00
edge_cnt                RMB     1       ;edge count
EDGE1                   RMB     2       ;captured first edge value
period                  RMB     2       ;period in number of E clock cycles
TEMP1                   RMB     2
FREQH                   RMB     2
DBUFR                   RMB     8

                ORG     $E8
                JMP     IC1_ISR

                ORG     $B600
                LDS     #$01FF          ;init stack
                LDX     #REGBAS
                CLRA
                STAA    EDGE1
                STAA    EDGE1+1
                JSR     IC1_init

LOOP            LDAA    #2              ;initialize edge_cnt to 2
                STAA    edge_cnt        ;
                CLI                     ;enable interrupts to 68HC11

                LDAB    #10             ;wait for 1 second
outer           LDY     #20000
inner           NOP
                NOP
                DEY
                BNE     inner
                DECB
                BNE     outer

WAIT            TST     edge_cnt        ;checks if two edges have been captured
                BNE     WAIT
                SEI                     ;disables interrupts
                CLR     period
                LDX     #REGBAS
                LDD     TIC1,X          ;get the second edge time
                SUBD    EDGE1           ;take the difference of edge1 and edge2
                STD     period
```

Figure 1: ASM code showcasing code controlling initialization of variables, one second delay, and loop used to attain period after two interrupts have occurred.

3. The two rising edges are recorded during the IC1 interrupt which occurs automatically without having to jump to its code. Figure 2 shows the code for the interrupt.

```
IC1_init         LDX     #REGBAS
                 LDAA    #IC1rise          ;prepare to capture the rising edge of IC1
                 STAA    TCTL2,X           ;         "
                 BCLR    TFLG1,X IC1FM     ;clear the IC1F flag of the TFLG1 register
                 LDAA    #2
                 STAA    edge_cnt
                 BSET    TMSK1,X IC1I      ;set the IC1 interrupt bit
                 RTS

IC1_ISR  LDX     #REGBAS
         BCLR    TFLG1,X IC1FM    ;clear the IC1F flag
         DEC     edge_cnt
         BEQ     SKIP             ;is this the second edge?
         LDD     TIC1,X
         STD     EDGE1            ;save the arrival time of the first edge
SKIP     RTI
```

Figure 2: ASM code showcasing code initialization of IC1 for rising edges and the IC1 ISR which records timestamps for period calculation

4. Once the period of a signal (in E-clock cycles) is determined, the program utilizes an algorithm which translates the period to frequency as a hexadecimal value. Figure 3 shows that once the frequency is determined, it is compared to a lower limit ($64 which is 100Hz) and an upper limit ($2710 which is 10kHz) and if it is out of range, displays a message that the frequency is too large or too small.

```
        LDD     #32             ;X-period D-32
        FDIV                    ;D/X -> X R->D
        STX     TEMP1

        LDD     TEMP1
        LSRD
        LSRD
        LSRD
        CLR     FREQH
        STAA    FREQH+1
        LSRD
        LSRD
        STD     TEMP1
        XGDX
        SUBD    TEMP1
        XGDX
        LSRD
        STD     TEMP1
        ADDA    FREQH+1
        STAA    FREQH+1
        XGDX
        SUBD    TEMP1
        ADDD    FREQH
        STD     FREQH

        CPD     #$0064          ;64(16) = 100(10) lower limit of freq
        BHS     OKP1            ;skip if okay
        LDX     #MSGER1
        JSR     .OUTSTR
        BRA     JTOP            ;else display too small

OKP1    CPD     #$2710          ;2710(16) = 10K(10) upper limit of freq
        BLS     OKP2            ;skip if okay
        LDX     #MSGER2
        JSR     .OUTSTR
        BRA     JTOP            ;else display too large

OKP2    JSR     .OUTCRL
        LDX     #FREQH
        JSR     HTOD
        JSR     P5DEC
        LDX     #MSGHZ
        JSR     .OUTSTO
JTOP    JMP     LOOP
```

Figure 3: ASM code showcasing code which converts period to frequency and displays value to the Buffalo monitor.

5. If the frequency is within the accepted range, a message is printed to the Buffalo monitor displaying the signal’s frequency in hertz. A subroutine is used to convert the previous frequency in hexadecimal to its corresponding decimal value, shown in figure 4.

```
HTOD    PSHX
        PSHB
        PSHA
        LDD     0,X
        LDX     #10000
        IDIV
        XGDX
        ADDB    #$30
        STAB    DBUFR
        XGDX
        LDX     #1000
        IDIV
        XGDX
        ADDB    #$30
        STAB    DBUFR+1
        XGDX
        LDX     #100
        IDIV
        XGDX
        ADDB    #$30
        STAB    DBUFR+2
        XGDX
        LDX     #10
        IDIV
        ADDB    #$30
        STAB    DBUFR+4
        XGDX
        ADDB    #$30
        STAB    DBUFR+3
        PULA
        PULB
        PULX
        RTS

MSGER1  FCC     "Freq. is below 100Hz"
        FCB     $04
MSGER2  FCC     "Freq. is above 10kHz"
        FCB     $04
MSGHZ   FCC     " Hz"
        FCB     $04
```

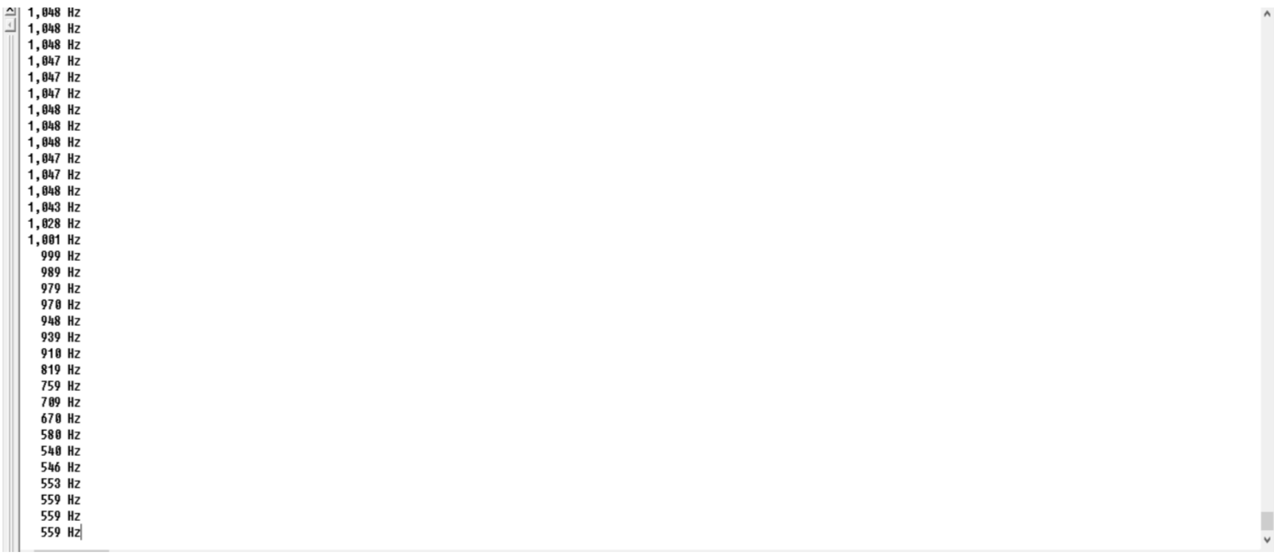
Figure 4: ASM code showcasing subroutine used to convert hexadecimal value to decimal frequency.

6. Once the frequency value is displayed, the program loops back to the second delay and waits for the next period to be captured. Three screen captures were obtained to show the program displaying correct results shown in figures 5, 6, and **7.**

Figure 5: Buffalo monitor displaying low frequencies and low frequency warning message.

```
 1,000 Hz
 1,000 Hz
 2,001 Hz
 2,001 Hz
 2,001 Hz
 2,998 Hz
 4,000 Hz
 5,000 Hz
 6,006 Hz
 7,999 Hz
10,000 Hz
Freq. is above 10kHz
Freq. is above 10kHz
Freq. is above 10kHz
Freq. is above 10kHz
Freq. is above 10kHz
Freq. is above 10kHz
Freq. is above 10kHz
Freq. is above 10kHz
Freq. is above 10kHz
 1,000 Hz
 1,000 Hz
```

Figure 6: Buffalo monitor displaying high frequencies and high frequency warning message.



```
1,048 Hz
1,048 Hz
1,048 Hz
1,047 Hz
1,047 Hz
1,047 Hz
1,048 Hz
1,048 Hz
1,048 Hz
1,047 Hz
1,047 Hz
1,048 Hz
1,043 Hz
1,028 Hz
1,001 Hz
  999 Hz
  989 Hz
  979 Hz
  970 Hz
  948 Hz
  939 Hz
  910 Hz
  819 Hz
  759 Hz
  709 Hz
  670 Hz
  580 Hz
  540 Hz
  546 Hz
  553 Hz
  559 Hz
  559 Hz
  559 Hz
```

Figure 7: Buffalo monitor displaying regular frequencies near neither the upper or lower limits of the range.

Conclusions

I encountered two large problems while completing the program for what should have been a relatively easy lab. Firstly, the textbook listed part of the original input capture example incorrectly. It showed that the IC1_init subroutine accessed the TCTL2 register directly instead of as an offset of REGBAS. (STAA TCTL2 instead of STAA TCTL2, X). I did not notice this typo and left it in my code so the IC1_ISR never activated. My second problem was encountered when trying to have the Buffalo monitor display frequency values. The first two digits of any frequency would display correctly, but any following digits would be shown incorrectly. I had forgotten to include a # sign in the subroutine which converts the frequencies from hexadecimal to decimal. Besides those problems, I enjoyed working on this lab.