**Design Assignment 5: Timer**

***Continuation of Design Assignment 4***

****

**Embedded Systems**

**ELC 411**

**Matthew Strickland and Jacob Levine**

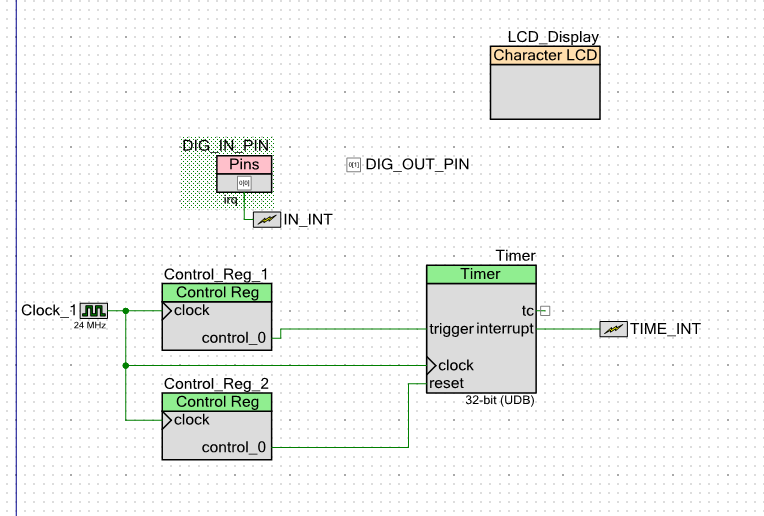
**Submission: 11/16/17**

1. **Diagrams**

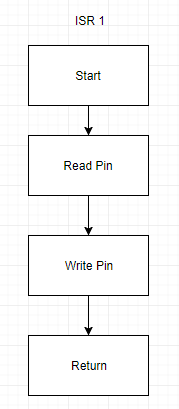
Figure 1 below shows the PSoC Schematic file for Design Assignment 5. In continuation of Design Assignment 4, the two control registers and a timer were added. The control registers interface with the timer. The trigger input pin enables the down counter (starting at its max value). When the timer reaches 0, the interrupt pin goes high which then causes the microprocessor to execute the TIME\_INT instruction (setting timer\_flag = 1). The input signal is read on a digital input pin which also has an interrupt. On each falling edge of the the input signal, the code within IN\_INT is executed.

Using timers instead of CyDelay allows for a more consistent delay time. Once the timer is enabled, the timer continues to decrement in parallel with any instructions the microprocessor is executing. Because this happens in parallel, a consistent delay is found. When using CyDelay, the interrupt service routine ‘steals’ clock cycles from the for loop which causes an inaccurate and inconsistent delay in time.

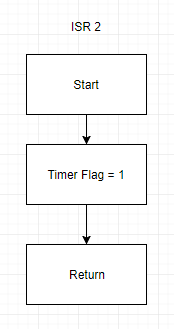
One major difference between this design assignment and the previous (in addition to fixed latency) is that the LCD did not have problems keeping up with the input frequency. Because the timer causes a fixed delay, and that delay is used to update the LCD, successful performance occurred until an input frequency of 197 kHz.



*Figure 1: PSoC Schematic*

**

*Figure 2: ISR 1 Block Diagram*

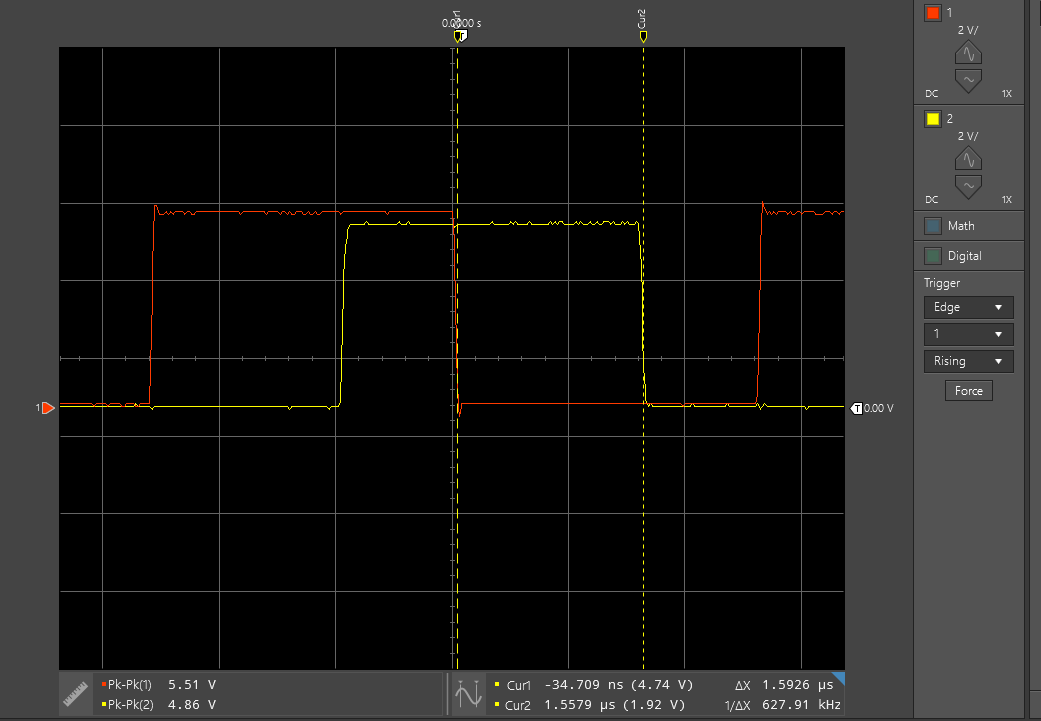
**

*Figure 3: ISR 2 Block Diagram*

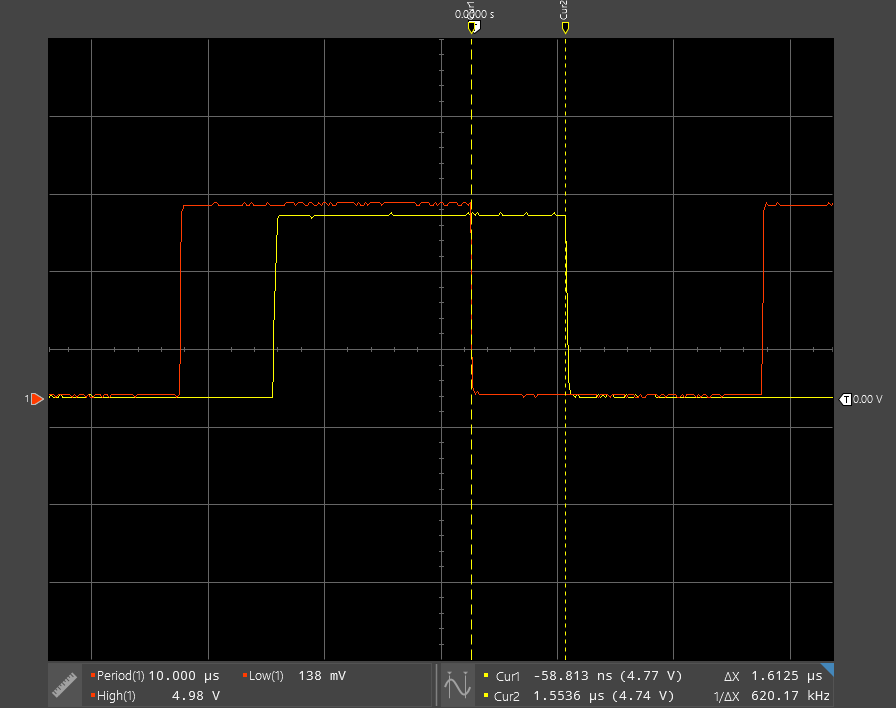
Figure 4 and 5 below show a consistent delay of around 1.6 uS regardless of input frequency. Using a clock rate of 24 MHz (as shown in Fig. 1 - PSoC schematic) this latency translates to around 38 clock cycles. The value of clock cycles was found using the following equation.

1.6E-6 \* (24E+6) = Clock Cycles

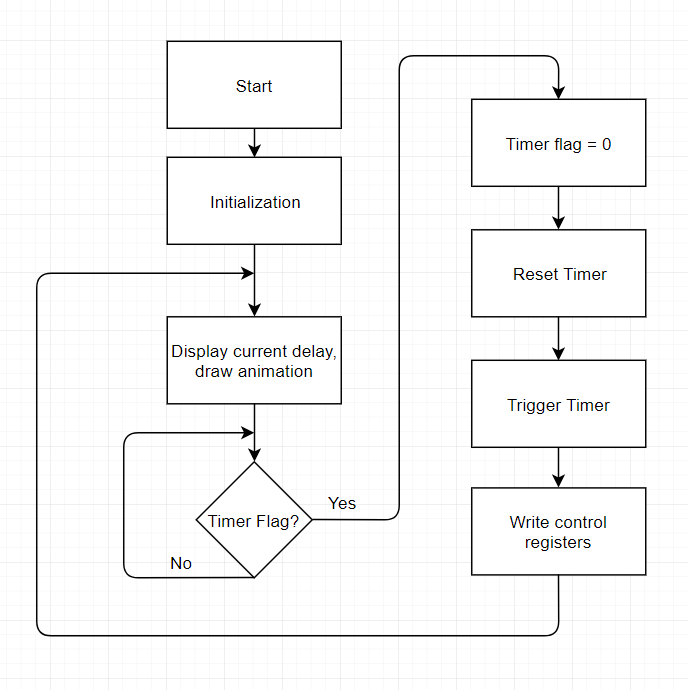
Clock Cycles = 38



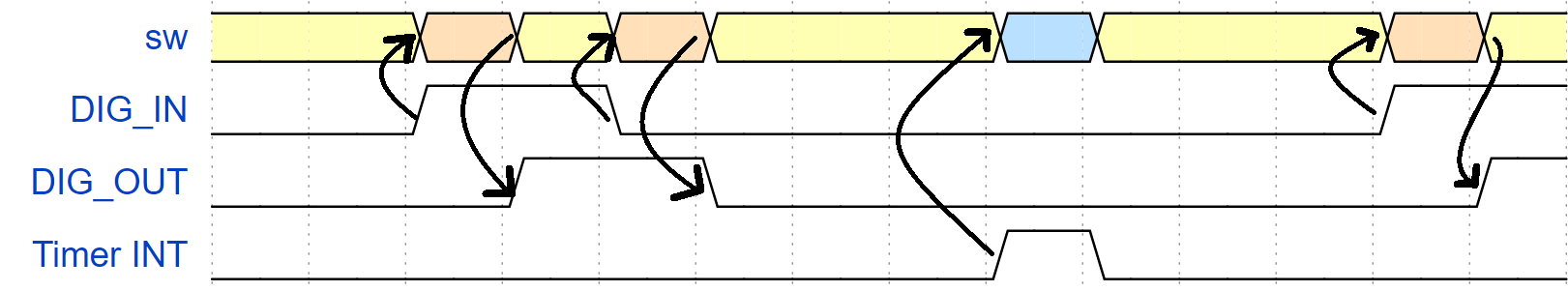
*Figure 4: Latency 197 kHz (Max Frequency)*



*Figure 5: Latency 100 kHz (Min Frequency)*

**

*Figure 6: Main System Block Diagram*

****

*Figure 7: Timing Diagram*

**Appendix:** Commented Source Code

#include <project.h>  
#include <stdio.h>  
#include <math.h>  
  
// Unsigned Fixed Point Macros, UQm.n  
#define FIX\_n (16) // fixed point 'n' value  
#define FIX\_m (16) // fixed point 'm' value  
#define FIX\_N (FIX\_n + FIX\_m) // total bits in UQm.n  
#define FIX\_FACTOR (1 << FIX\_n ) // fixed point fraction factor (2^n)  
#define FIX\_\_0\_5 (1 << (FIX\_n-1)) // 0.5 expressed in UQ16.16  
#define FIX\_\_1\_0 (1 << FIX\_n ) // 1.0 expressed in UQ16.16  
  
static int volatile timer\_flag;  
  
CY\_ISR ( DIG\_IN\_PIN\_HANDLER )   
{  
 DIG\_OUT\_PIN\_Write(DIG\_IN\_PIN\_Read());   
 DIG\_IN\_PIN\_ClearInterrupt();  
}   
  
CY\_ISR ( TIMER\_HANDLER )  
{  
 timer\_flag = 1;  
}   
  
// Procedure:  
// fix2int - round a fixed point value to the nearest integer  
// Inputs:  
// fix - fixed point value to round  
// Return value:  
// result of rounding the fixed point value to nearst integer, in uint32\_t container  
  
uint32\_t fix2int( uint32\_t fix )  
{  
 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 uint32\_t container;   
 container = (fix + FIX\_\_0\_5) >> FIX\_n; //   
 return container;  
 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
}  
  
// Procedure:  
// fix2double - Convert a fixed point value to double precision   
// Inputs:  
// fix - value to convert  
// Return value:  
// double precision representation of fixed point value  
double fix2double( uint32\_t fix )  
{  
 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 double doubleContainer;  
 doubleContainer = (double)fix/FIX\_FACTOR;  
 return doubleContainer;  
 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
}  
  
// Procedure:  
// double2fix - Convert a double precision value to fixed point, with rounding  
// Inputs:  
// x - value to convert  
// Return value:  
// fixed point approximation of the input value, in uint32\_t container  
uint32\_t double2fix( double x )  
{  
 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 uint32\_t container;  
 container = (uint32\_t)((x \* FIX\_FACTOR) + 0.5);  
 return container;  
  
 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
}  
  
// Procedure:  
// fix2decimalstr - convert fixed point value to decimal string  
// Inputs:  
// x - fixed point value to convert to string  
// str - pointer to destination string  
// dotn - desired decimal precision  
void fix2decimalstr( uint32\_t x, char \*str, int dotn )  
{  
 int i;  
 uint64\_t lx;  
 int len;  
   
 // Use 64 bit int to avoid overflow  
 lx = x;  
   
 // Multiply by 10^dotn, to shift all fractional decimal into integer part  
 for (i = 0; i < dotn; ++i)  
 {  
 lx \*= 10;  
 }  
 // Get the integer part via rounding by adding half, and right shifting n  
 lx += FIX\_\_0\_5;  
 lx >>= FIX\_n;  
   
 x = (uint32\_t) lx;  
   
 // Print the number, but without decimal point  
 sprintf( str, "%d", (int) x );  
 len = strlen(str);  
   
 // Insert the decimal point in the correct location  
 // First move all of the last 'dotn' characters to the right to make space  
 str[len+1] = '\0';  
 for (i = 0; i < dotn; ++i)  
 {  
 str[len-i] = str[len-i-1];  
 }  
 str[len-dotn] = '.';  
}  
  
int main()  
{  
 int k; // Current position of bouncing box (relative to LCD)  
 int direction; // +1 --> move right, -1 --> move left  
 char num\_str[17]; // Array to render the value of rate as a string  
 char msg\_str[17]; // Entire message, to write to LCD  
   
 int sw2; // Holds current switch state  
 int sw3; // Holds current switch state  
 int sw2\_prev; // Holds previous state, for button down detection  
 int sw3\_prev; // Holds previous state, for button down detection  
   
 CyGlobalIntEnable; /\* Enable global interrupts. \*/  
 LCD\_Display\_Start();  
 IN\_INT\_StartEx( DIG\_IN\_PIN\_HANDLER );  
 TIME\_INT\_StartEx( TIMER\_HANDLER );  
 timer\_flag = 0;  
 Timer\_Start();  
 timer\_flag = 0;  
 Control\_Reg\_2\_Write(1);  
 Control\_Reg\_1\_Write(1);  
   
 uint32\_t delay = 20 \* FIX\_\_1\_0; // UQ16.16  
 uint32\_t llim = 20 \* FIX\_\_1\_0; // Upper limit of delay expressed in UQ16.16  
 uint32\_t ulim = 200 \* FIX\_\_1\_0; // Upper limit of delay expressed in UQ16.16  
  
 uint32\_t incr = double2fix( 10.0/3.0 ); // Represent 3.33... in fixed point  
   
 // Start the LCD component  
   
 k = 0; // Initialize position  
 direction = 1; // and direction  
   
 sw2 = sw3 = sw2\_prev = sw3\_prev = 1; // Initialize switch states to open  
   
 // Loop forever  
 for(;;)  
 {  
 // Convert current delay to a string, with 3 decimal places precision  
 fix2decimalstr(delay, num\_str, 3);  
   
 // Generate composite message string  
 sprintf( msg\_str, "Delay=%7s ms", num\_str );  
  
 // Render current state onto the display  
 // Top line is bouncing square  
 // Bottom line is current delay  
 LCD\_Display\_ClearDisplay(); // Must clear entire display before new rendering  
 LCD\_Display\_DrawHorizontalBG(0, k, 1, 5); // Draw the box on top line  
 LCD\_Display\_Position(1, 0); // Position on bottom line  
 LCD\_Display\_PrintString(msg\_str); // Print the msg on bottom line  
  
 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
 //DIG\_OUT\_PIN\_Write(DIG\_IN\_PIN\_Read());  
 // \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
   
   
 while (timer\_flag == 0)  
 {  
 // do nothing until timer flag is set to 1   
 }   
   
 timer\_flag = 0;  
 Control\_Reg\_2\_Write(1);  
 Control\_Reg\_1\_Write(1);  
   
 //CyDelay(fix2int(delay)); // Delay based on integer part of delay  
 k += direction; // Compute new position  
 if (k == 15)  
 direction = -1;  
 else if (k == 0)  
 direction = 1;  
  
 sw2 = SW2\_Read(); // Get current switch state  
 //sw3 = SW3\_Read(); // Get current switch state  
   
 if (sw2 == 0 && sw2\_prev == 1) // If Switch 2 button down event, decrease delay  
 delay -= incr;  
 if (sw3 == 0 && sw3\_prev == 1) // If Switch 3 button down event, increase delay  
 delay += incr;  
 sw2\_prev = sw2; // Update previous sw2 state  
 sw3\_prev = sw3; // Update previous sw3 state  
  
 // Saturate delay to upper and lower limits  
 if (delay > ulim) delay = ulim;  
 if (delay < llim) delay = llim;  
 }  
}  
  
/\* [] END OF FILE \*/

|  |  |  |  |
| --- | --- | --- | --- |
| **Item** | **Expected** | **Points** | **Pts. Available** |
| Cover sheet |  | 0.5 | 0.5 |
| Flowchart - foreground thread | Init - forever { stuff, wait until timer, rearm timer, stuff) | 1 | 1 |
| Flowchart - DIG IN ISR thread | Read DIG IN, write DIG out, clear pin interrupt | 1 | 1 |
| Flowchart - Timer ISR thread | set timer\_flag | 1 | 1 |
| Table - measurements of latency | at sq. wave freq. of 100 KHz, 180 KHz | 0.5 | 0.5 |
| Description of animation speed vs. clock rate | Animation speed is virtually constant regardless of clock rate, until system fails | 0.5 | 0.5 |
| Screen shot one waveform, overlaid with reference lines showing latency measurement | With volts/div, us/div | 1 | 1 |
| What time delay, using equations? | 33 ms | 0.5 | 1 |
| Draw timing diagram |  | 2.5 | 2.5 |
| main.c file, two ISR files, fully commented and strictly formatted |  | 1 | 1 |
| **TOTAL** |  | **9.5** | **10** |