```
// Justin Maeder & Zachery Takkesh
// PLL at 80MHz
// UART 7 (PE0-1) for receiving frequency values from first TM4C123
// Port B pins 0-7 for outputting sine wave to frequency
//      Using 32 steps to create sine wave 262-494Hz with formula.

// 1. Pre-processor Directives Section
// Constant declarations to access port registers using
// symbolic names instead of addresses

#include "SysTick.h"
#include "SysTick.c"
#include "stdio.h"
#include "stdlib.h"
#include "tm4c123gh6pm.h"
#include "PLL.h"
#include "PLL.c"
#include <math.h>


#define PI                              3.14159265358979323846
#define SYSCTL_RCGCUART_R7      0x00000080  // UART Module 7 Run Mode Clock
#define CR   0x0D
#define LF   0x0A
#define BS   0x08

// 2. Declarations Section
//   Global Variables
unsigned short mode;
unsigned char data;
unsigned short i,j;
unsigned long k;
char temp;
unsigned int frequency, nfrequency;
//unsigned char data;

void DisableInterrupts(void);
void EnableInterrupts(void);

void sine(void);
void OutCRLF(void);

unsigned char UART_InCharNonBlocking(void);
unsigned char UART7_NonBlockingInChar(void);

unsigned char UART_InChar(void);
unsigned char UART7_InChar(void);

void UART_OutChar(unsigned char data);
void UART7_OutChar(unsigned char data);

unsigned long UART_InUDec(void);
unsigned long UART7_InUDec(void);

void UART_OutString(char *pt);
void UART_OutUDec(unsigned long n);

void UART7_Init(void);
void PortA_Init(void);
void PortB_Init(void);

// 3. Subroutines Section
// MAIN: Mandatory for a C Program to be executable
int main(void){
  DisableInterrupts();
    PLL_Init();
  UART7_Init();
    PortA_Init();
    PortB_Init();
```

```
70    EnableInterrupts();
71      mode = 1;
72      GPIO_PORTB_DATA_R = 0x00;
73      OutCRLF();
74      frequency = 262;
75      nfrequency = 0;
76    while(1){
77          //UART_OutChar('a');
78          frequency = UART7_InUDec();
79
80
81          if(nfrequency != frequency){
82
83              OutCRLF();
84              UART_OutUDec(frequency);
85              k = 2500000 / frequency;
86              DisableInterrupts();
87              mode = 1;
88              //UART_OutUDec(k);
89              //GPIO_PORTB_DATA_R  = 0x7F;
90              SysTick_Init(k);
91              EnableInterrupts();
92              nfrequency = frequency;
93          }
94          sine();
95          //UART_OutChar('d');
96      } // End while
97  } // End main
98
99
100 void sine(void){
101         for(i = 0; i < 32; i++){           // 32 outputs because 256 has too much
            overhead
102             j = (sin(PI*(i/16.0))+1) * 128;  // Simplified sine equation
103             if(j > 255) j--;                 // Limit 255 because index 64 rounds up to
                256
104             GPIO_PORTB_DATA_R = j;
105             SysTick_Wait(65000);
106     }
107 }
108
109 void OutCRLF(void){
110   UART7_OutChar(CR);
111   UART7_OutChar(LF);
112 }
113
114 unsigned char UART_InCharNonBlocking(void){
115   if((UART0_FR_R&UART_FR_RXFE) == 0){
116     return((unsigned char)(UART0_DR_R&0xFF));
117   } else{
118     return 0;
119   }
120 }
121
122 unsigned char UART7_NonBlockingInChar(void){
123     if((UART7_FR_R&UART_FR_RXFE)==0)
124         return((unsigned char)(UART7_DR_R&0xFF));
125     else
126         return 0;
127 }
128
129 unsigned char UART_InChar(void){
130   while((UART0_FR_R&UART_FR_RXFE) != 0);
131   return((unsigned char)(UART0_DR_R&0xFF));
132 }
133
134 unsigned char UART7_InChar(void){
135   while((UART7_FR_R&UART_FR_RXFE) != 0);
136   return((unsigned char)(UART7_DR_R&0xFF));
```

```
}

void UART_OutChar(unsigned char data){
  while((UART0_FR_R&UART_FR_TXFF) != 0);
  UART0_DR_R = data;
}

void UART7_OutChar(unsigned char data){
  while((UART7_FR_R&UART_FR_TXFF) != 0);
  UART7_DR_R = data;
}


/*
unsigned long UART_InUDec(void){

unsigned long number=0, length=0;
char character;
  character = UART_InCharNonBlocking();
  while(character != CR){
        UART_OutChar('d'); // PROBLEM NOT ABLE TO ESCAPE
    if((character>='0') && (character<='9')) {
      number = 10*number+(character-'0');
      length++;
      UART_OutChar(character);
    }
    else if((character==BS) && length){
      number /= 10;
      length--;
      UART_OutChar(character);
    }

    character = UART_InCharNonBlocking();
  }
  return number;
}


unsigned long UART7_InUDec(void){
unsigned long number=0, length=0;
char character;
  character = UART7_NonBlockingInChar();
  while(character != CR){
    if((character>='0') && (character<='9')) {
      number = 10*number+(character-'0');
      length++;
      UART7_OutChar(character);
    }
    else if((character==BS) && length){
      number /= 10;
      length--;
      UART7_OutChar(character);
    }
        UART_OutUDec(number);
    character = UART7_NonBlockingInChar();
  }
    UART_OutString("FUCK");

  return number;
}
*/

unsigned long UART_InUDec(void){
unsigned long number=0, length=0;
char character;
  character = UART_InChar();
  while(character != CR){ // accepts until <enter> is typed
// The next line checks that the input is a digit, 0-9.
// If the character is not 0-9, it is ignored and not echoed
```

```
206       if((character>='0') && (character<='9')) {
207         number = 10*number+(character-'0');   // this line overflows if above 4294967295
208         length++;
209         UART_OutChar(character);
210       }
211   // If the input is a backspace, then the return number is
212   // changed and a backspace is outputted to the screen
213       else if((character==BS) && length){
214         number /= 10;
215         length--;
216         UART_OutChar(character);
217       }
218       character = UART_InChar();
219     }
220     return number;
221   }
222
223   unsigned long UART7_InUDec(void){
224   unsigned long number=0, length=0;
225   char character;
226     character = UART7_InChar();
227       //UART_OutChar('Z');
228     while(character != CR){ // accepts until <enter> is typed
229   // The next line checks that the input is a digit, 0-9.
230   // If the character is not 0-9, it is ignored and not echoed
231       if((character>='0') && (character<='9')) {
232         number = 10*number+(character-'0');   // this line overflows if above 4294967295
233         length++;
234               UART7_OutChar(character);
235           }
236   // If the input is a backspace, then the return number is
237   // changed and a backspace is outputted to the screen
238       else if((character==BS) && length){
239         number /= 10;
240         length--;
241               UART_OutChar('C');
242         UART7_OutChar(character);
243       }
244
245       character = UART7_InChar();
246
247     }
248       //UART_OutUDec(number);
249
250     return number;
251   }
252
253
254   void UART_OutString(char *pt){
255     while(*pt){
256       UART_OutChar(*pt);
257       pt++;
258     }
259   }
260
261   void UART_OutUDec(unsigned long n){
262     if(n >= 10){
263       UART_OutUDec(n/10);
264       n = n%10;
265     }
266     UART_OutChar(n+'0'); /* n is between 0 and 9 */
267   }
268
269   void UART7_Init(void){
270     SYSCTL_RCGCUART_R |= 0x80; // activate UART7
271     SYSCTL_RCGCGPIO_R |= 0x10;
272     SYSCTL_RCGC2_R    |= 0x10; // activate port E
273
274     UART7_CTL_R    &= ~UART_CTL_UARTEN;   // disable UART
```

```
275       UART7_IBRD_R    = 86;                 // IBRD, 80Mhz clk, 57600 |
          int(80,000,000/(16*57600)) = int(86.8055556)
276       UART7_FBRD_R    = 52;                 // FBRD | int(64*0.8055556+0.5) = 52.055552
277                                             // 8 bit word length (no parity bits, one stop
                                                bit, FIFOs)
278     UART7_LCRH_R = 0x0070;
279     UART7_CTL_R |= UART_CTL_UARTEN;         // enable UART
280     GPIO_PORTE_AFSEL_R |= 0x03;             // enable alt funct on PE1-0
281     GPIO_PORTE_DEN_R   |= 0x03;             // enable digital I/O on PE1-0
282                                             // configure PE1-0 as UART
283     GPIO_PORTE_PCTL_R   = (GPIO_PORTE_PCTL_R&0xFFFFFF00)+0x00000011;
284     GPIO_PORTE_AMSEL_R &= ~0x03;
285   }
286
287   void PortA_Init(void){
288     SYSCTL_RCGC1_R |= SYSCTL_RCGC1_UART0; // activate UART0
289     SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOA; // activate port A
290     UART0_CTL_R &= ~UART_CTL_UARTEN;      // disable UART
291     UART0_IBRD_R = 86;                   // IBRD, 80Mhz clk, 38400 |
          int(80,000,000/(16*38,400)) = int(130.2083)
292     UART0_FBRD_R = 52;                    // FBRD | int(0.2083*64+0.5) = 13.8312
293                                           // 8 bit word length (no parity bits, one stop
                                              bit, FIFOs)
294     UART0_LCRH_R = 0x0070;
295
296     UART0_CTL_R |= UART_CTL_UARTEN;       // enable UART
297     GPIO_PORTA_AFSEL_R |= 0x03;           // enable alt funct on PA1-0
298     GPIO_PORTA_DEN_R |= 0x03;             // enable digital I/O on PA1-0
299                                           // configure PA1-0 as UART
300     GPIO_PORTA_PCTL_R = (GPIO_PORTA_PCTL_R&0xFFFFFF00)+0x00000011;
301     GPIO_PORTA_AMSEL_R &= ~0x03;          // disable analog functionality on PA
302   }
303
304   void PortB_Init(void){
305       volatile unsigned long delay;
306       SYSCTL_RCGC2_R |= 0x00000002;
307       delay = SYSCTL_RCGC2_R;
308       GPIO_PORTB_LOCK_R = 0x4C4F434B;
309       GPIO_PORTB_CR_R    |=  0xFF;
310       GPIO_PORTB_AMSEL_R &= ~0xFF;
311       GPIO_PORTB_DIR_R   |=  0xFF;  //PB0-7 output
312       GPIO_PORTB_AFSEL_R &= ~0xFF;  //Regular I/O
313       GPIO_PORTB_PCTL_R  &= ~0xFF;  //GPIO
314       GPIO_PORTB_PUR_R   &= ~0xFF;  //no pull-up res
315       GPIO_PORTB_DEN_R   |=  0xFF;
316   }
317
```