Reporte Laboratorio #7 Microcontroladores

Emanuel Lascurain A01552126
Victor Cavazos A01177055

Procedimientos

Usamos nuestro código de la práctica 6 como base para empezar a desplegar en nuestro display LCD como lo es requerido, por lo cual en nuestro código tenemos las funciones LCD_rdy, LCD_cmd, LCD_init y send2LCD con la cual desplegamos nuestros valores de frecuencia.
Para calcular nuestra frecuencia, tuvimos que probar varias veces el uso de nuestro TIMER 0 y 1 y así poder detectar la frecuencia en Hz empleada.
Lo logramos con una simple funcion de if, primero para detectar nuestra interrupt flag en PIEbits.TMR1IE && PIR1bits.TMR1IF, de esta manera sabemos que detectamos un pulso de voltaje y aqui encendemos nuestro TIMER0 para que empiece a contar, despues implementamos una serie de if para determinar el tiempo que estuvo encendido y asi desplegar en el display cual fue la frecuencia, esto en un ciclo while para que pueda estar cambiando cada segundo.

```
#include "device_config.h"
#define _XTAL_FREQ 8000000
#define SWEEP_FREQ 50
#define LCD_RS              LATCbits.LATC2
#define LCD_RS_DIR          TRISCbits.TRISC2
#define LCD_RW              LATCbits.LATC1
#define LCD_RW_DIR          TRISCbits.TRISC1
#define LCD_E               LATCbits.LATC0
#define LCD_E_DIR           TRISCbits.TRISC0
#define LCD_DATA_R          PORTD
#define LCD_DATA_W          LATD
#define LCD_DATA_DIR        TRISD
unsigned char counter_t0 = 0;
unsigned char counter_100ms = 0;
//unsigned char counter_t0 = 0;
//unsigned char counter_t0 = 0;
enum por_ACDC {digital, analog};

char key_scanner(void);
char is_number(char number);
char is_sign(char sign);
char is_enter(char enter);
char is_ac(char ac);

void LCD_init(void);
void LCD_cmd(char);
void send2LCD(char);
```

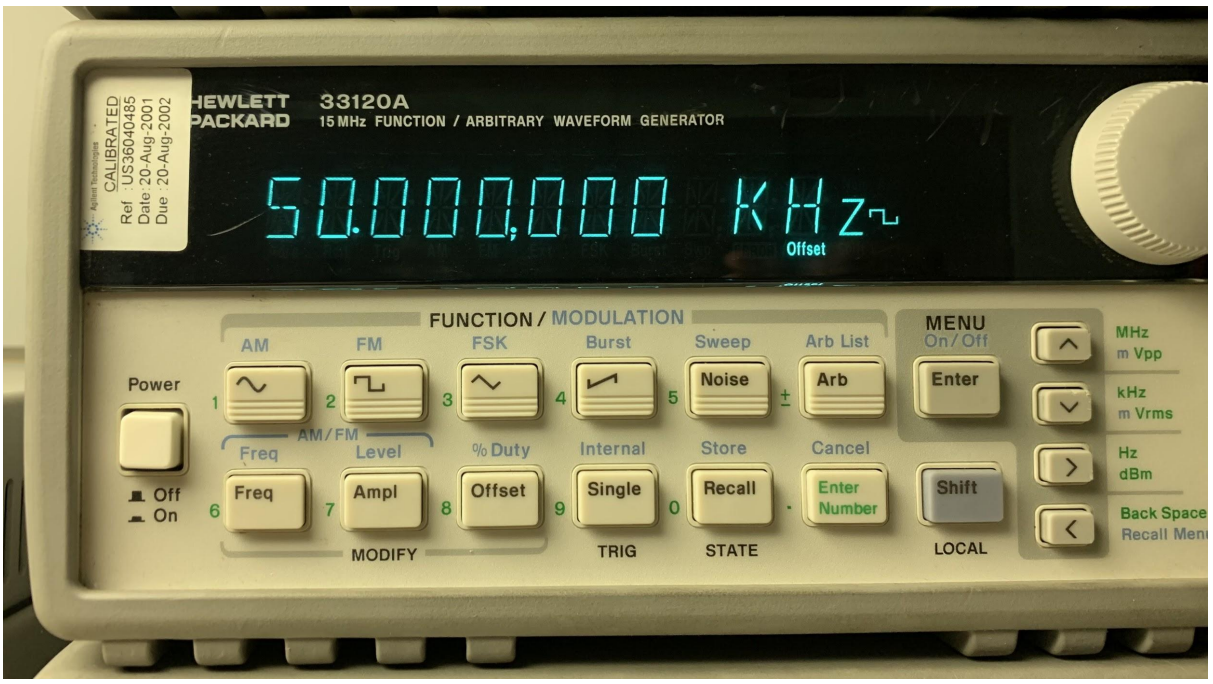
```
void LCD_rdy(void);
void portsInit(void);

void main(void){
   OSCCON = 0x64;
   INTCONbits.GIE = 0;
   T1CONbits.RD16 = 1;
   T1CONbits.TMR1CS = 0;
   T1CONbits.T1CKPS = 0b11;
   TMR1H = 0x9E;
   TMR1L = 0x57;
   TRISAbits.TRISA4=1;
   TRISB=0;
   T0CON=0x68;
   TMR0L=0;
   LCD_DATA_DIR = 0x00;
   LCD_RS = 0;
   LCD_RW = 0;
   LCD_E  = 0;
   LCD_init();

   send2LCD('F');
   send2LCD('R');
   send2LCD('E');
   send2LCD('C');
   send2LCD('U');
   send2LCD('E');
   send2LCD('N');
   send2LCD('C');
   send2LCD('I');
   send2LCD('A');
   send2LCD(':');
   LCD_cmd(0xCB);
   __delay_ms(25);
   send2LCD('H');
   send2LCD('z');
   T1CONbits.TMR1ON = 1;
   PIR1bits.TMR1IF = 0;
   PIE1bits.TMR1IE = 1;
   INTCONbits.PEIE = 1;
   INTCONbits.GIE = 1;
   T0CONbits.TMR0ON=1;

   while(1);
}
void send2LCD(char xy){
   LCD_RS = 1;
   LCD_RW = 0;
```

```
    LCD_E  = 1;
    LCD_DATA_W = xy;
    Nop();
    Nop();
    LCD_E  = 0;
    __delay_ms(250);
}

void LCD_init(void){
    LATC = 0;               // Make sure LCD control port is low
    LCD_E_DIR = 0;          // Set Enable as output
    LCD_RS_DIR = 0;         // Set RS as output
    LCD_RW_DIR = 0;         // Set R/W as output
    LCD_cmd(0x38);          // Display to 2x16
    __delay_ms(250);
    LCD_cmd(0x0F);          // Display on, cursor on and blinking
    __delay_ms(250);
    LCD_cmd(0x01);          // Clear display and move cursor home
    __delay_ms(250);
}

void LCD_rdy(void){
    char test;

    LCD_DATA_DIR = 0xFF;
    test = 0x80;
    while(test){
        LCD_RS = 0;
        LCD_RW = 1;
        LCD_E  = 1;
        test = LCD_DATA_R;
        Nop();
        LCD_E = 0;
        test &= 0x80;
    }
}

void LCD_cmd(char cx) {
    //LCD_rdy();
    LCD_RS = 0;
    LCD_RW = 0;
    LCD_E  = 1;
    __delay_ms(25);
    LCD_DATA_W = cx;
    __delay_ms(25);
    LCD_E = 0;
}
```

```
void __interrupt () ISR_TIMER_1(void)
{
   if (PIE1bits.TMR1IE && PIR1bits.TMR1IF)
   {
      T0CONbits.TMR0ON=1;
      if (++counter_100ms > 9) {
         LCD_cmd(0xC0);
         __delay_ms(30);
         if(TMR0L < 10){
            send2LCD('0' + TMR0L);
         }
         if(TMR0L >= 10 && TMR0L < 100){
            send2LCD('0' + TMR0L/10);
            send2LCD('0' + TMR0L%10);
         }
         if(TMR0L >= 100 && TMR0L < 255){
            send2LCD('0' + TMR0L/100);
            send2LCD('0' + (TMR0L%100)/10);
            send2LCD('0' + ((TMR0L%100)/10)%10);
         }
         counter_100ms = 0;
         TMR0 = 0;
         T0CONbits.TMR0ON=0;
         INTCONbits.TMR0IF=0;
      }
      PIR1bits.TMR1IF = 0;
    }
  }
```

Resultados

Haciendo uso de nuestro programa se puso a prueba con el generador de frecuencia, con un VPP de 5V, un offset de 2.5v y una Frecuencia de 50kHz para este ejemplo.


CALIBRATED
Ref : US36040485
Date : 20-Aug-2001
Due : 20-Aug-2002
HEWLETT PACKARD
33120A
15 MHz FUNCTION / ARBITRARY WAVEFORM GENERATOR
50.000,000 KHz
Offset
FUNCTION / MODULATION
AM
FM
FSK
Burst
Sweep
Arb List
Noise
Arb
MENU
On / Off
Enter
Power
Off
On
AM / FM
Freq
Level
% Duty
Internal
Store
Cancel
Freq
Ampl
Offset
Single
Recall
Enter Number
Shift
MODIFY
TRIG
STATE
LOCAL
MHz
m Vpp
kHz
m Vrms
Hz
dBm
Back Space
Recall Menu

Pese a que el valor final no fue exactamente el mismo que el de la señal cuadrada generada, consideramos se acerca lo suficiente para considerar que la práctica funciona.

Conclusiones

Emanuel: Pese a que fue una práctica bastante compleja debido a el uso de timers, al fin pudimos completarla y ejecutarla de manera satisfactoria, personalmente me quedo con muchas ganas de seguir aprendiendo sobre cómo utilizar la curiosity.

Victor: En esta practica entendi mas los conceptos teoricos vistos en clase sobre el uso de TIMERS y mas que nada su configuracion y como afecta tanto saber leer el manual y ver todos los bits.