Instituto Politécnico Nacional

Ingeniería en Sistemas Computacionales

Laboratorio de Instrumentación

Practica N° 8

Protocolo RS-232

Alumno: _____

Boleta: _____ Grupo: _____

Profesor: _____

Fecha de elaboración: ___/___/___.

Protocolo RS-232

Objetivo

El alumno aprenderá a emplear el protocolo RS-232, así como a identificar las diferentes etapas que lo constituyen, para de esta manera contar con un medio de comunicación serial, con en el cual se puedan comunicar un microcontrolador con una PC.

Equipo empleado

- ✓ 1 Multímetros
- ✓ 1 Fuente de VCD variable
- ✓ 1 Osciloscopio
- ✓ 2 Puntas Banana – Caimán
- ✓ 2 Puntas de osciloscopio
- ✓ Protoboard
- ✓ PIC16F628A
- ✓ 3 Resistencia 390 Ω @ ¼ Watt
- ✓ 8 Resistencia 10 KΩ @ ¼ Watt
- ✓ 2 Capacitores 0.1 µF
- ✓ 1 Regulador 7805
- ✓ 1 Dipswitch de 8 interruptores
- ✓ 1 Led rojo
- ✓ 1 Led verde
- ✓ 1 Led amarillo

Desarrollo de la práctica

1.- Circuito generador de señal bajo protocolo RS-232.

Arme el circuito de la figura 1. Utilice +V = 6 a 12 VCD.

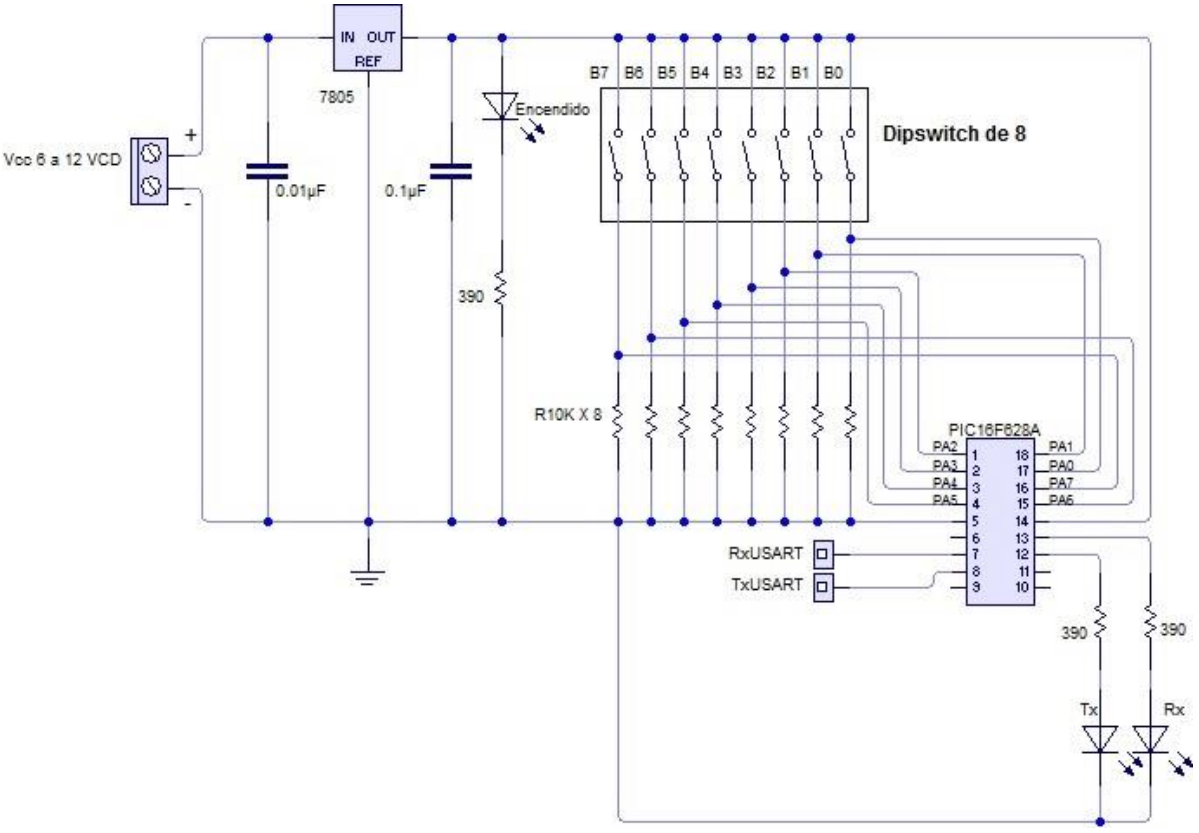

Figura 1

Al microcontrolador se le tiene que grabar el código (el código del programa se encuentra en al final del presente documento) previamente ensamblado en el cual, las terminales del puerto A, deben configurarse como entradas, para que lean el estado lógico que será generado por el dipswitch. Para ello se podrán fijas las 8 terminales del dipswitch en la posición que se quiera.

Posteriormente, el microcontrolador después de leer las terminales del puerto A, enviara la información de cada uno de estos bits, por medio de la salida Tx que trabajara mediante protocolo RS-232. Para visualizar el tren de datos será necesario conectar el osciloscopio, en la terminal identificada como TxUSART (la terminal positiva del cable de osciloscopio a la terminal TxUSART, y el negativo a tierra).

Cada segundo el microcontrolador estará enviando el tren de datos que se está generando por medio del dipswitch, y para cambiarlo se tiene que modificar la posición de las terminales del dipswitch.

A continuación, observe la señal que se genera en el osciloscopio, y dibújela en el espacio que se encuentra a continuación.

	B7	B6	B5	B4	B3	B2	B1	B0
Byte que se fijó en el dipswitch								

Cambie la combinación binaria que se encuentra en el dipswitch y de nueva cuenta, dibuje la señal en el espacio siguiente.

	B7	B6	B5	B4	B3	B2	B1	B0
Byte que se fijó en el dipswitch								

Con las señales anteriores calcule la tasa de transferencia (bits por segundo), y coloque su valor a continuación.

Baud Rate = _____KBaud.

2.- Cambio del valor de la tasa de transferencia.

Se tendrá que reprogramar al microcontrolador, por lo que se tiene que apagar la fuente de alimentación, sacar el microcontrolador, y proceder a realizar el cambio en el programa.

El cambio en el programa consiste en lo siguiente: Ubique donde se encuentra la instrucción mediante la cual se carga un valor al registro “spbrg”, y cambiar valor que se tiene de 25 (decimal) por el de 12 (decimal).

Programe de nueva cuenta al microcontrolador, instálelo en el circuito, encienda la fuente de voltaje y vuelva a fijar las combinaciones binarias que anteriormente se tenían, y vuelva a obtener las señales llenando los espacios que se encuentran a continuación.

	B7	B6	B5	B4	B3	B2	B1	B0
Byte que se fijó en el dipswitch								

	B7	B6	B5	B4	B3	B2	B1	B0
Byte que se fijó en el dipswitch								

De estas nuevas señales anteriores calcule la tasa de transferencia (bits por segundo), y coloque su valor a continuación.

Baud Rate = _____KBaud.

Cuestionario

1. Identifique las diferentes partes que componen a la señal RS-232 que se obtuvo en la práctica.

2. ¿Qué significa tasa de transferencia?

3. Menciona 5 aplicaciones que se le pueden asignar al protocolo RS-232.

4. ¿Cuál es el rango de voltaje que entrega una PC en el puerto serie, bajo protocolo RS-232?

Conclusiones

Anote las conclusiones a las que llego con el desarrollo de esta práctica.

Anexo Código para programar al microcontrolador

A continuación, esta expresado el código para programar al microcontrolador, mismo que puede copiar al editor en lenguaje C, para generar el archivo hexadecimal (.hex).

```
// CONFIG
#pragma config FOSC = INTOSCIO    // Oscillator Selection bits (INTOSC oscillator: I/O function on RA6/OSC2/CLKOUT pin, I/O function on RA7/OSC1/CLKIN)
#pragma config WDTE = OFF         // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = ON         // Power-up Timer Enable bit (PWRT enabled)
#pragma config MCLRE = OFF        // RA5/MCLR/VPP Pin Function Select bit (RA5/MCLR/VPP pin function is digital input, MCLR internally tied to VDD)
#pragma config BOREN = OFF        // Brown-out Detect Enable bit (BOD disabled)
#pragma config LVP = OFF          // Low-Voltage Programming Enable bit (RB4/PGM pin has digital I/O function, HV on MCLR must be used for programming)
#pragma config CPD = OFF          // Data EE Memory Code Protection bit (Data memory code protection off)
#pragma config CP = OFF           // Flash Program Memory Code Protection bit (Code protection off)

// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>
#include <pic16f628a.h>

#define _XTAL_FREQ 4000000

#define Bit7 PORTAbits.RA7
#define Bit6 PORTAbits.RA6
#define Bit5 PORTAbits.RA5
#define Bit4 PORTAbits.RA4
#define Bit3 PORTAbits.RA3
#define Bit2 PORTAbits.RA2
#define Bit1 PORTAbits.RA1
#define Bit0 PORTAbits.RA0

#define RxUART PORTBbits.RB1
#define TxUART PORTBbits.RB2
#define Led_Tx PORTBbits.RB6
#define Led_Rx PORTBbits.RB7

#define GIE INTCONbits.GIE
#define TOIE INTCONbits.T0IE
#define TOIF INTCONbits.T0IF
#define PEIE INTCONbits.PEIE

#define SPEN RCSTAbits.SPEN
#define CREN RCSTAbits.CREN

#define TRMT TXSTAbits.TRMT

#define RCIE PIE1bits.RCIE
#define TXIE PIE1bits.TXIE

#define TXIF PIR1bits.TXIF
#define RCIF PIR1bits.RCIF

char ByteRx;
char TxUSART;

void __interrupt() VectorInterrupcion(void)        // Funciones de interrupción.
{
   if (RCIF == 1)                                  //Interrupción por uso de UART (RS-232)
            {  GIE = 0;                            //Desactivación general de interrupciones
               ByteRx = RCREG;
               RCIF = 0;                           //Limpia la bandera de interrupción por recepción
               GIE = 1;                            //Activa las interrupciones
            }
}

void ConfigPIC(void)
{
   TRISA = 0B11111111;                             //Puerto A como entradas
   TRISB = 0B00000010;                             //Puerto B como salidas solo bit 1 como entrada
   CMCON = 0X07;                                   //Terminales del puerto A como I/O digitales
   PORTA = 0;
   PORTB = 0;
}

void ConfigUSART(void)                             //Configuracion_USART  9600 BPS, sin bit de paridad, 1 bit stop
{
   TXIE = 0;                                       //Desactiva interrupción por fin de transmisión por usart.
   TXSTA = 0B00100110;
   SPBRG = 25;
   RCSTA = 0B10010000;
   //Este bit es parte del registro RCSTA   SPEN =1;          //Habilitación del puerto de comunicación serial
   //Este bit es parte del registro RCSTA   CREN = 1;         //Activa la recepción continua
```

```
    RCIE = 1;                           //Activar interrupción por fin de recepción por usart.
    TXIF = 0;
    RCIF = 0;
    GIE = 1;                            //Activar habilitador general de interrupciones.
    PEIE = 1;                           //Activar habilitador general de interrupciones por periféricos.
    RCREG = 0;
}

void Transmite(void)                    //Transmite por RS-232 9600 BPS, sin bit de paridad, 1 bit stop
{
    TXSTA = 0B00100110;
    SPBRG = 25;
    TXREG = TxUSART;
    while(!TXIF);
    while(!TRMT);                       //Esta bandera es la que funciona para detectar cuando se vacía el buffer de transmisión
}

void main(void)
{
    ConfigPIC();
    ConfigUSART();

    while (1)
    {   TxUSART = PORTA;
        Led_Tx = 1;
        Transmite();
        __delay_us(100);
        Led_Tx = 0;
        __delay_ms(1000);
    }
    return;
}
```