Comunicación serie Maestro/Esclavo

Laura Porcel Martín, Juan Carlos Gutiérrez Baños.

Práctica de la asignatura Microordenadores

lauraporma@gmail.com
gutierrezjuanca@hotmail.com

Resumen — **En este trabajo se ha implementado un dispositivo de comunicación serie en configuración MAESTRO/ESCLAVO. Se ha utilizado el microcontrolador PIC16F876 presente en el laboratorio y el programa de control RealPic. El objetivo ha sido el diseño de un protocolo a nivel de capa red sobre un nivel físico basado en el estándar RS232 en su versión asíncrona.**

I. Introducción

Una comunicación Maestro/Esclavo requiere un conjunto de reglas que especifiquen el intercambio de datos u órdenes. Estas reglas definen lo que se conoce como un protocolo de red o también un protocolo de comunicación. En este caso el intercambio se produce entre dos microcontroladores PIC16F876.

Este microcontrolador utiliza a nivel físico el protocolo RS232 encargado de establecer una norma para el intercambio serie de datos binarios entre un equipo terminal de datos y un equipo de Comunicación de datos. La interfaz RS232 está diseñada para distancias cortas, de unos 15 metros o menos, y para velocidades de comunicación bajas, de no más de 20 [Kb/s]. La interfaz puede trabajar en comunicación asíncrona o síncrona y tipos de canal simplex, half duplex o full duplex. En este caso será asíncrona y utilizará un canal simplex en el que el intercambio de datos se realiza en un solo sentido.

A. Objetivos

En este trabajo se han configurado dos entrenadores basados en el PIC16F876 para establecer una comunicación mediante un periférico (USART) que utiliza el protocolo RS232.

Se han implementado tanto una etapa de conexión como una etapa de comunicación de datos entre los entrenadores. La etapa de conexión permite determinar si el entrenador, configurado como esclavo está presente. Este proceso consiste en el intercambio de caracteres específicos. La etapa de comunicación permite al entrenador configurado como maestro enviar los caracteres introducidos en su teclado y al entrenador esclavo recibirlos y mostrarlos en su pantalla LCD. Véase la Figura 1 una fotografía del entrenador con los periféricos utilizados.

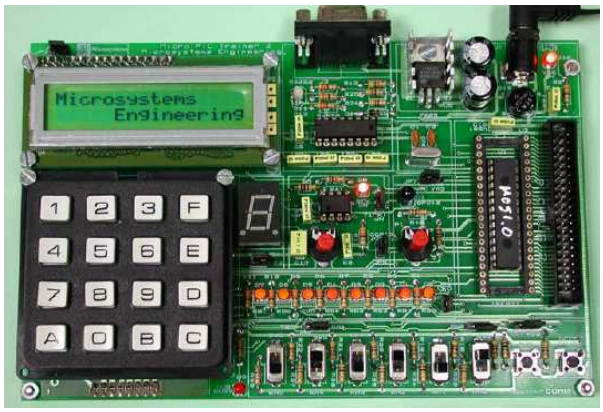

Fig. 1 Fotografía del entrenador utilizado (PIC 16F876).

II. Implementación Hardware

B. Periférico USART.

El periférico USART es el utilizado para la transmisión de datos en formato serie, aplicando técnicas de transmisión sincrónica o asincronica, según su configuración. La característica más destacable de este periférico es que destina un terminal a la transmisión (Tx) y otro a la recepción (Rx), en este caso el sincronismo se hace dentro de cada equipo y la interfaz solo define el uso de un bit de start y otro de stop, para indicar el inicio y fin de transmisión de un byte, es por eso que todos los equipos interconactados deben estar configurados para el mismo bit-rate. Las ventajas más importantes de este modo de comunicación radica en que no se requiere destinar más entradas salidas a completar algunas interfaces como la RS232.

C. Implementación del cable de conexión

Los dos entrenadores se conectan entre ellos cruzando los pines del puerto RS232 de recepción y transmisión y uniendo los pines de tierra. El pin encargado de recibir es el que ocupa la posición 2, el pin de transmisión ocupa la posición 3 y el pin de tierra o gnd es el número 5.

Véase Figura 2 en la que se muestra el esquema básico de la conexión entre dos entrenadores PIC16F876.

Fig. 2 Esquema de la implementación.

III. IMPLEMENTACIÓN SOFTWARE

El entrenador distinguirá entre los dos modos de operación según la posición del interruptor RA4, nivel bajo para Maestro y nivel alto para Esclavo.

A. Comprobación inicial sobre el interruptor RA4.

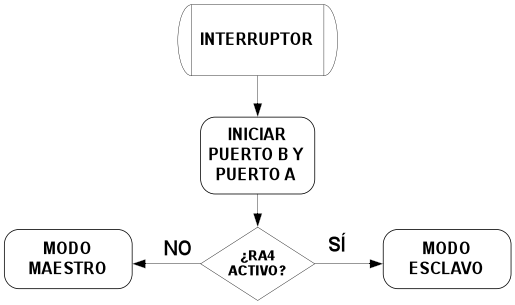

Fig. 3 Diagrama de flujo para seleccionar el modo.

Nuestro sistema de comunicación Maestro/Esclavo se inicia consultando el modo en el que queremos trabajar para una vez decidido comenzar nuestra transmisión de datos. El bucle con que se inicia el programa principal es, en código ensamblador, el siguiente:

```
Loop:       movf    PORTA,W
            andlw   b'00010000'
            btfsc   STATUS,Z
            goto    Vale_0
            movwf   interruptor
            movlw   b'00000000'
            btfsc   STATUS,Z
            goto    Vale_1
Vale_1
            goto InicioEsclavo
Vale_0
            goto InicioMaster
```

B. Modo Maestro.

Inicializaremos este modo con un bucle de petición de conexión, en el que el entrenador enviará cada dos segundos el carácter '@' a través del canal TX a no ser que reciba '=' por su canal RX. Mientras estemos en este bucle el LCD mostrará por pantalla el siguiente mensaje: '…Detectando…'.

Una vez establecida la comunicación, el entrenador enviará por su canal TX el dato introducido en el teclado y mostrará por pantalla el dato enviado y el mensaje '…Conectado…'

Seguimos el siguiente diagrama de flujo:

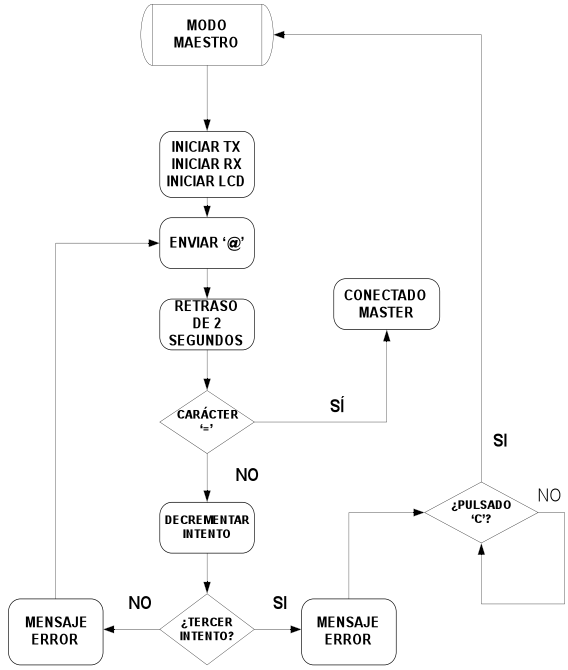

Fig. 4 Diagrama de flujo Modo Maestro.

En código ensamblador quedaría de la siguiente forma:

```
InicioMaster:   call inicio_tx
                call inicio_rx
                call inicio_lcd
                call conectando_lcd
                movlw   0xc5
                call    LCD_REG
                call master_lcd
                movlw  .3
                movwf  intentos

LoopConexionMaster:

                call enviar_@
                call esperar2seg
                call capturar
                call comparar_igual
                call intento
                goto    LoopConexionMaster
```

Utilizando la llamada a subrutinas hemos conseguido construir un programa fácil de entender. A tener en cuenta, destacamos que la función *comparar_igual* nos hará salir del bucle de conexión cuando recibamos el carácter '=', dirigiéndonos al estado conectado el cual veremos a continuación, y la subrutina *intento* al llegar al tercer intento entraremos a un bucle de espera pidiendo reintentar la conexión.

Subrutina c*onectado Maestro e Intento:*

```
ConectadoMaster:
                call    LCD_INI
                movlw   0x80
                call    LCD_REG
                movlw   Mens_1
                 call    Mensaje
                goto enviar
                return
```

```
Intento
        call    error_lcd
        call    esperar1seg
        movlw   0xc5
        call    LCD_REG
        call    master_lcd
        decfsz  intentos
        return
        goto FalloConexion
```

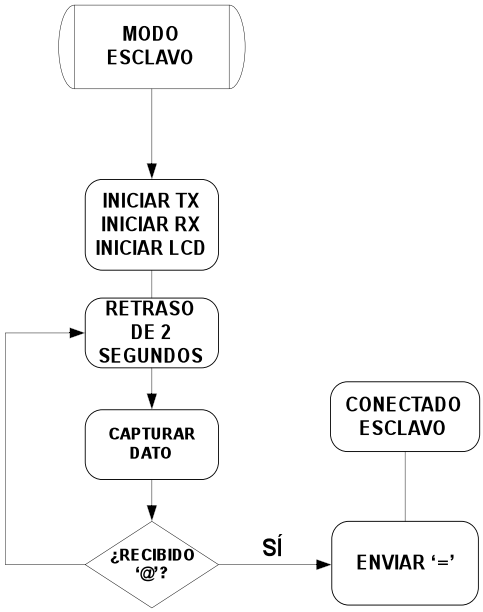

Fig. 5 Diagrama de flujo Modo Esclavo.

C. *Modo Esclavo*

El entrenador, configurado como Esclavo, iniciará un bucle de espera de carácter '@' y mostrará por el LCD el mensaje '…Conectando…'

Una vez conectados ambos entrenadores, nuestro entrenador mostrará por pantalla el valor que reciba por el canal RX además del mensaje '…Conectado…'

Su correspondiente diagrama de flujo se puede observar en la Fig. 5.

Y su código en ensamblador, también con un programa inicial fácil de entender, es:

```
InicioEsclavo:
        call    inicio_tx
        call    inicio_rx
        call    inicio_lcd
        call    esperar_lcd
        movlw   0xc5
        call    LCD_REG
        call    esclavo_lcd
LoopConexionEsclavo:
        call    esperar2seg
        call    capturar
        call    comparar_@
        goto    LoopConexionEsclavo
```

La subrutina *capturar* consultamos el dato recibido en el registro RCREG y lo introducimos en la variable global RECIBIDO.

B. *Parte opcional realizada*

Como elemento opcional añadimos a nuestro sistema de comunicación control de errores en la conexión, limitando este a tres intentos, tras el cual si no conseguimos conectarnos pediremos pulsar la tecla 'C' para reintentar conectarnos.

IV. Conclusiones

En este artículo hemos desarrollado un software capaz de interconectar dos PIC16F87, teniendo en cuenta una fase preliminar de petición de conexión, no es un sistema muy complejo de realizar y sus utilidades en este caso también son escasas, aunque si cabe destacar que es la base principal para poder implementar un software basado en controladores PIC con una capacidad de comunicación muy superior.

Referencias

[34] Microchip PIC16F87X Data Sheet, 28/40-Pin 8-Bit CMOS FLASH Microcontrollers.
[35] Tomeu Alorda, Apuntes de la asignatura de Microcontroladores, de 2º Telemática, UIB.
[36] Mikel Etxebarria, (c) Microsystems Engineering (Bilbao), ejemplos Real Pic.
[37] William Stalings, "Comunicaciones y redes de computadores", 6ª ed., Pearson Education, 2000.

Asignatura impartida por Bartomeu Alorda y Pere Pons

Juan Carlos Gutiérrez Baños
Bachillerato en Colegio San Pedro.
Estudiante de 3º de Ingeniería Técnica de Telecomunicaciones esp. Telemática.

Laura Porcel Martín
Bachillerato en I.E.S. Guillem Sagrera.
Estudiante de 3º de Ingeniería Técnica de Telecomunicaciones esp. Telemática.