Polibits

ISSN: 1870-9044

polibits@nlp.cic.ipn.mx

Instituto Politécnico Nacional

México

Herrera Lozada, Juan Carlos; Tlizaliturri Flores, Ian; Morales Castillo, Mario

Unidad Básica de Comunicación Serial en un Microcontrolador

Polibits, núm. 33, 2006, pp. 3-7

Instituto Politécnico Nacional

Distrito Federal, México

Unidad Básica de Comunicación Serial en un Microcontrolador

M. en C. Juan Carlos Herrera Lozada
Investigador del CIDETEC IPN
Ian Iizaliturri Flores
Alumno de la Maestría en Tecnología de Cómputo del CIDETEC IPN
Mario Morales Castillo
Alumno de la Maestría en Tecnología Avanzada del CICATA IPN (Legaria)

El presente artículo infiere el diseño de una unidad de comunicación serial implementada con un microcontrolador. La funcionalidad de este prototipo permite enviar y recibir datos de cualquier sistema de cómputo que soporte el estándar RS-232, dado que su generalidad permite adaptarlo a cualquier aplicación sin cambios drásticos. Este trabajo expone los resultados preliminares del proyecto *"Sistemas de Control para Motores, Supervisados por Computadoras de Bolsillo"* que se desarrolla actualmente en el CIDETEC - IPN.

Introducción

Los microcontroladores actuales normalmente cuentan con un módulo funcional de comunicación serie ***USART/SCI*** *(Universal Synchronous Asynchronous Receiver Transmitter/Serial Comunication Interface);* en caso contrario éste puede implementarse en la mayoría de sus funciones mediante software. Este módulo se utiliza ampliamente en sistemas de cómputo en general, por lo que el microcontrolador visto como un *core* o unidad básica puede establecer comunicación de manera simple con el procesador del equipo y acceder a los recursos del sistema completo.

El medio de transmisión para la interfaz *USART* requiere de al menos dos vías, por lo que la complejidad, costos y el uso de terminales del microcontrolador se reducen; esta interfaz tiene dos modos de comunicación: asíncrona o síncrona. La primera es *full duplex* con una vía RX (Recepción) y otra TX (Transmisión), la segunda interfaz es *half duplex*, utiliza una vía de CLK (reloj) y otra de DT (Transmisión de datos), con sus configuraciones por dispositivo como maestro o esclavo. En los microcontroladores también pueden existir otros módulos de comunicación serie como el *SPI (Serial Peripherial Interface)* o *I2C (Inter.-Integrated Circuit)* que son de tipo síncrono.

Cuando el módulo *USART* maneja la comunicación en modo asíncrono se denomina ***UART*** *(Universal Asynchronous Receiver/Transmitter).* Para establecer comunicación mediante el *UART* se requiere que los parámetros sean iguales para TX y RX; estos parámetros son: número de bits del dato, número de bits de paro *(stop)*, velocidad de transmisión (en baudios), control de paridad y niveles lógicos utilizados. La línea que transmite los datos en serie está inicialmente en estado alto; al comenzar la transferencia se envía un bit "0" lógico o *bit de inicio*. Tras él irán los bits de datos a transmitir (en ocasiones son 8, 7, 6 o 5): estos bits están espaciados con un intervalo temporal fijo y preciso, ligado a la velocidad de transmisión que se esté empleando.

Posteriormente podría venir un bit de paridad que indica si se ha enviado un número par o impar de bits con un nivel lógico alto en la palabra; al final aparecerá un bit (a veces un bit y medio ó dos bits) a "1" lógico, representando los bits de paro. Medio bit significa que la señal correspondiente en el tiempo a un bit dura la mitad; realmente, en comunicaciones se utiliza el término baudio para hacer referencia a las velocidades y por lo general, un baudio equivale a un bit por segundo. La presencia de los bits de inicio y parada permite sincronizar la estación emisora con la receptora, haciendo que los relojes de ambas vayan a la par, por eso el tipo de transmisión se denomina asíncrona por paquete y síncrona por bit.

Existen varios estándares de comunicación serie asíncrona, que emplean tensiones diferenciadas apropiadas para distancias largas entre dispositivos como RS-422, RS-485 u otros que emplean niveles de tensión para la inmunidad a ruido, tal como RS-232. El *UART* tiene compatibilidad con este último en su tercera versión, denominada con "C".

Estándar Rs-232c

Propuesto por la *Asociación de Industrias Electrónicas (EIA)*, realizándose posteriormente una versión internacional por el ***CCITT*** *(Consultative Committee for International Telegraphy and Telephony)* conocida como V.24 con pocas diferencias entre ambas. El estándar estipula las características mecánicas, eléctricas y de un protocolo orientado al enlace físico punto a punto. Las características eléctricas ofrecen alta inmunidad a ruido, y una distancia de enlace de hasta 16 metros dependiendo de la velocidad de transmisión utilizada. Los niveles de voltaje aplicado para el 0 lógico están en el rango de +3V a +15V, y para el 1 lógico se tiene un rango de -3V a -15V; de -3V a +3V se tiene una zona de transición. Este estándar cuenta con dos tipos de conectores físicos: el conector DB9 y el conector DB25. La **Tabla 1** muestra la configuración física de las terminales para ambos casos.

Diseño del Prototipo

El *core* diseñado está dispuesto como enlace entre el sistema de cómputo y una aplicación externa (monitoreo de señales, adquisición de datos, sistemas de control, etc.). El microcontrolador intercambia información con el equipo de cómputo vía el puerto serie. Por ejemplo, una PC común utiliza circuitos integrados *UART* tales como el 8250, el 16450, el 16550, etc.; cada uno de estos con sus variantes en máxima velocidad de comunicación, elementos de almacenamiento, control de flujo de datos, funcionalidad adicional y en el cumplimiento de la norma RS-232C; por ello, adecuar los requerimientos de la unidad básica de comunicación serial (microcontrolador programado como tal) no representa mayor problema.

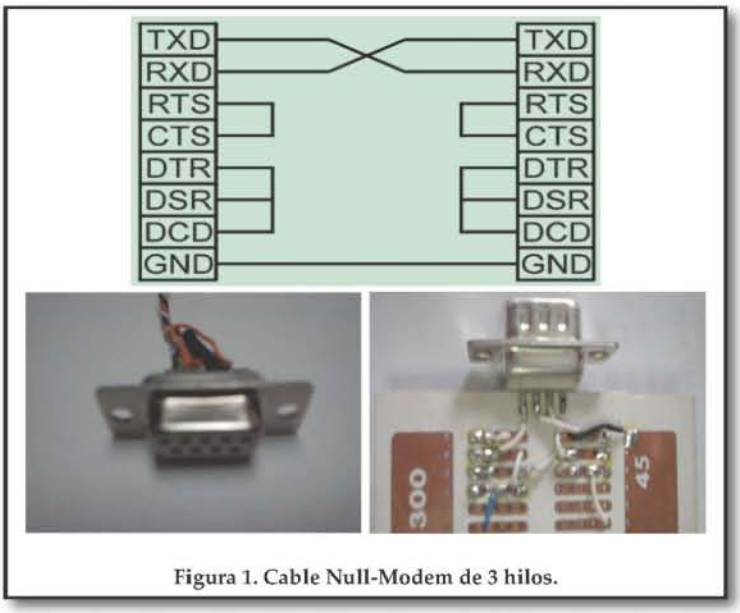

Figura 1. Cable Null-Modem de 3 hilos.

Construcción del Cable

Para la comunicación serial simplificada, considerando la robustez que pueda presentar el diseño, se requiere un cable *Null – Modem* de 3 hilos, interconectando las señales sobrantes en el mismo conector DB9, tal y como se aprecia en la **Figura 1**. Este procedimiento emula el protocolo CTS/RTS y DSR/DTR por hardware; para controlar el flujo de datos se recurre al protocolo de software XON/XOFF.

Pin DB9	Pin DB25	Nombre	Dirección	Descripción
1	8	CD	Entrada	Detección de portadora
2	3	RxD	Entrada	Recepción de datos
3	2	TxD	Salida	Transmisión de datos
4	20	DTR	Salida	Terminal de datos preparadas
5	7	GND	-	Masa del sistema
6	6	DSR	Entrada	Set de datos preparados
7	4	RTS	Salida	Petición para enviar
8	5	CTS	Entrada	Listo para enviar
9	22	RI	Entrada	Indicador de llamada

Tabla 1. Pines de los conectores DB9 y DB25.

Unidad Serial Implementada en Microcontrolador

Es posible utilizar cualquier microcontrolador considerando algunas excepciones que se mencionan posteriormente. Para unificar la metodología se utilizó primeramente el microcontrolador *PIC16F628A* de *Microchip*, debido a que es un dispositivo de bajo costo y que se consigue fácilmente en el mercado nacional; las características de este circuito integrado son: 2048x14 bits en memoria de código, 224x8 bits en memoria de datos, 128x8 bits en memoria EEPROM, una frecuencia de trabajo máxima de 20MHz, 16 puertos E/S, dos comparadores analógicos, *USART* y un módulo PWM de 10 bits. El módulo *USART* incluido tiene una memoria FIFO, ***BRG*** *(Baud Rate Generador)* y control de paridad. Se recomienda acceder a la hoja de especificaciones de este microcontrolador para profundizar en el mismo.

Programación en PicBasic Pro

Para la programación del microcontrolador se utiliza ***PICBASIC PRO,*** el cual utiliza un lenguaje de alto nivel que disminuye considerablemente la complejidad y el tiempo de diseño.

símbolo	valor	baudios	modos
T2400	0	2400	TTL Lógica Positiva
T1200	1	1200	
T9600	2	9600†	
T300	3	300	
N2400	4	2400	TTL Lógica Invertida
N1200	5	1200	
N9600	6	9600†	
N300	7	300	
OT2400	8	2400	Drenaje Abierto
OT1200	9	1200	
OT9600	10	9600†	
OT300	11	300	
ON2400	12	2400	Fuente Abierta
ON1200	13	1200	
ON9600	14	9600†	
ON300	15	300	

Tabla 2. Modos de transmisión aceptados por SERIN y SEROUT.

Este lenguaje contiene dos instrucciones básicas para la comunicación serial bajo un protocolo simple: ***SERIN y SEROUT*** (comunicación serial asíncrona para entrada/salida). El compilador, así como el manual de usuario y algunos tutoriales, se pueden obtener directamente del sitio web del fabricante *http://www.melabs.com/pbpdemo.htm*.

La sintaxis de la instrucción de entrada serial *SERIN* es la siguiente:

```
SERIN Pin, Mode, {Timeout,Label,}{[Qual...],}
{Item...}
```

En ella se indica la recepción de datos sobre la terminal (pin) indicada (de acuerdo a la distribución de pines del microcontrolador) en el formato estándar asíncrono que utiliza 8 bits de datos, sin bit de paridad y un solo bit de paro *(8N1)*. El modo selecciona la frecuencia de transmisión y el tipo de la misma de acuerdo a la **Tabla 2**. ***Mode*** (*p.e. T2400* para indicar el baudaje) está definido en el archivo *MODEDEFS.BAS*, que se incluye utilizando la línea *Include "modedefs.bas"*

La sintaxis de la instrucción para la salida serial *SEROUT* es:

```
SEROUT Pin, Mode,[Item[,Item...]]
```

Esta instrucción envía los datos de manera serial a través del pin indicado, utilizando el mismo formato estándar de la instrucción *SERIN*. Para utilizar el protocolo RS-232 completo con la interfaz *USART* (no sólo tres hilos, sino todos los del conector), se utilizan *HSERIN* y *HSEROUT*. Es este caso será necesario definir los parámetros de la comunicación y construir el cable adecuado.

Aplicación con Comunicación Completa

El circuito de la **Figura 2** muestra al microcontrolador PIC16F628A conectado a un motor CD. El bloque denominado *TERMINAL* representa a la interfaz de usuario, que permite enviar datos hacia el microcontrolador y visualizar los datos recibidos. Esta terminal puede ser propietaria, o es posible diseñarla con Visual Basic, C, o algún otro lenguaje que permita acceso al puerto serie. En este trabajo se utilizaron dos terminales, una para simulación (*contenida en la herramienta software ISIS de Proteus*) y otra para pruebas físicas que se descargó de *http://bray.velenje.cx/avr/terminal/*.

El código programado destina tiempo de proceso para establecer parámetros iniciales de funcionamiento que se estipulan en correspondencia al manejo de una *USART* en protocolo completo. En este circuito, la velocidad de giro y el paro del motor pueden manejarse en cualquier momento mediante la terminal. Obsérvese que el control del motor se realiza a través de la instrucción *HPWM* que permite modular el ancho del pulso y obtener un equivalente analógico en voltaje para reducir o aumentar la velocidad del motor.

El circuito integrado *MAX232* permite acoplar los voltajes de la *USART* y el PIC (opción más recomendable), aunque los microcontroladores de *Microchip* permiten acoplar los voltajes sin necesidad de un dispositivo adicional.

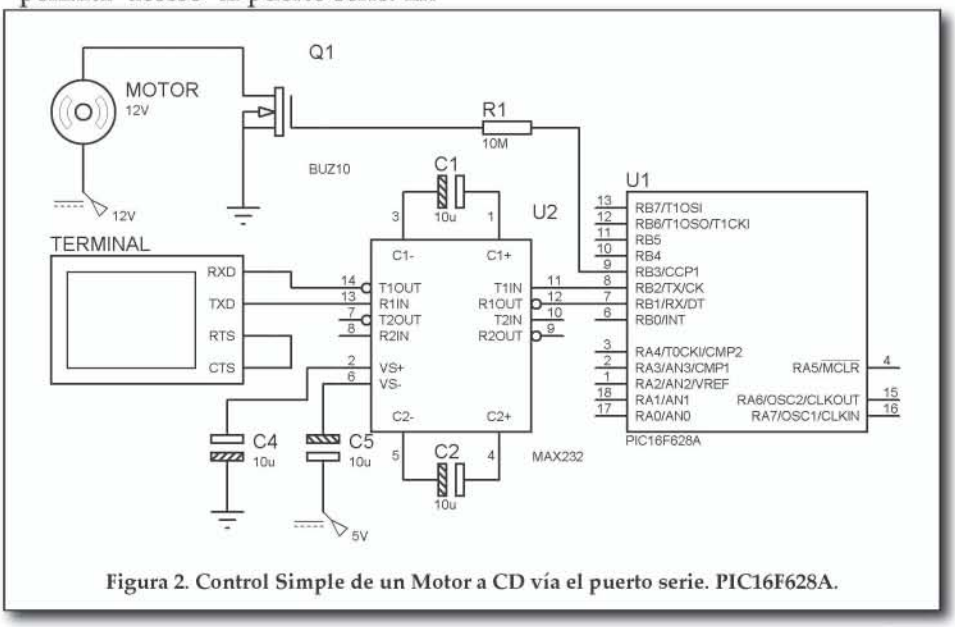

Figura 2. Control Simple de un Motor a CD vía el puerto serie. PIC16F628A.

```
Include "modedefs.bas"
'BITS DE CONFIGURACION DEL MICRO-
CONTROLADOR
@ DEVICE MCLR_OFF, INTRC_OSC,
WDT_OFF,
@ DEVICE LVP_OFF, BOD_OFF, PWRT_
OFF, PROTECT_OFF

'CONFIGURACION DE TERMINALES
DEFINE HSER_BAUD 2400    'VELOCI-
DAD
DEFINE HSER_SPBRG 25
DEFINE HSER_BITS 8    'BITS DE DATOS

'DECLARACION DE VARIABLES
RESPUESTA VAR BYTE

'PROGRAMA PRINCIPAL
HSEROUT ["CONTROL DE VELOCIDAD
PARA MOTOR DC",10,13]
INICIO: HSEROUT ["INTRODUZCA OPCIÓN
DE VELOCIDAD 1 2 3, STOP 0",10,13]
HSERIN [str RESPUESTA\1\"A"]
HSEROUT [RESPUESTA,10,13]
SELECT CASE RESPUESTA
CASE 49    'Velocidad 1
 HPWM 1,85,245
case 50    'Velocidad 2
 HPWM 1,170,245
case 51    'Velocidad total
 HPWM 1,255,245
case 48    'Paro de motor
 HPWM 1,0,245
end select
GOTO INICIO
END
```

Aplicación con Comunicación Simplificada

El microcontrolador utilizado en el circuito de la **Figura 3** es un *PIC12C508A*, que cuenta con 768x12 bits en memoria de código, 25x8 bits en memoria de datos, frecuencia máxima de 4MHz, 6 líneas de E/S y 8 pines. Este dispositivo no tiene módulos *PWM* y *USART*, por lo que su funcionamiento debe ser programado en su totalidad y tomando tiempo de procesamiento para su utilización. En el caso de la función *PWM*, sólo se utiliza para poder procesar la siguiente acción; en la comunicación no se utilizó el circuito de acoplamiento de voltajes, por lo que el estado de transición en 0v o menos en RS-232C es tomado en el microcontrolador como "0" lógico y cualquier voltaje arriba de 5v es tomado como "1" lógico; en el RS-232 los 0v del microcontrolador son tomados como "1" lógico y los 5v equivalen a "0". El no adecuar el acoplamiento de voltajes aumenta los errores y reduce la velocidad en la comunicación.

```
Include "modedefs.bas"
@ DEVICE MCLR_OFF,INTRC_OSC,WDT_
OFF,PROTECT_OFF

'DECLARACIÓN DE VARIABLES
RESPUESTA var byte
VELOCIDAD VAR BYTE
TIEMPO VAR BYTE

'PROGRAMA PRINCIPAL
VELOCIDAD = 0
SEROUT GPIO.0,N2400,["CONTROL DE
VELOCIDAD PARA MOTOR DC",10,13]
INICIO: SEROUT GPIO.0,N2400,["DEME
LA VELOCIDAD 1 2 3 PARO POR TIEM-
PO",10,13]

SERIN GPIO.1,N2400,[],RESPUESTA'Espe
ra para respuesta
SEROUT GPIO.0,N2400,[RESPUESTA,1
0,13]
SELECT CASE RESPUESTA
CASE 49    'Velocidad 1
 VELOCIDAD = 85
case 50    'Velocidad 2
 VELOCIDAD = 179
case 51    'Velocidad total
 VELOCIDAD = 255
end select
 FOR TIEMPO = 1 TO 5  ' M i c r o
4MHz,5ms*255*5 = 6 seg
 PWM GPIO.5,VELOCIDAD,255
 NEXT TIEMPO
 PWM GPIO.5,0,1
SEROUT GPIO.0,N2400,["FIN DE
PWM",10,13]
GOTO INICIO
END
```

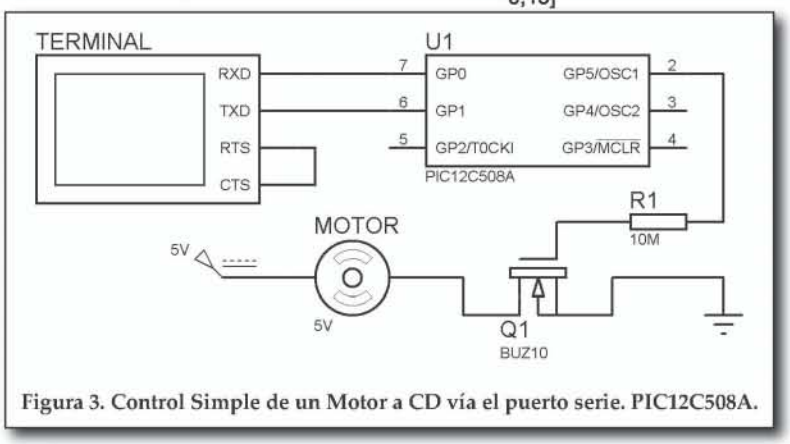

Figura 3. Control Simple de un Motor a CD vía el puerto serie. PIC12C508A.

Conclusiones

La transmisión serial resulta fácil de implementar, además de que es segura y de bajo costo. En el caso de un microcontrolador como interfaz hardware de intercambio con cualquier sistema de cómputo (principalmente PC's, y computadoras de bolsillo), es viable considerar las propuestas realizadas en este trabajo, ya que se adaptan a cualquier aplicación similar, debido a que la unidad básica de comunicación serie es la misma en todo momento.

Se mostraron dos aplicaciones que utilizan el *core* diseñado; por un lado un microcontrolador de la gama media con un dispositivo adicional para el acoplamiento de voltajes para la transmisión; por otro, un microcontrolador de la gama baja y sin acoplamiento de voltajes, con lo que se tiene una referencia mayor para realizar otras aproximaciones (ver **Figura 4**).

La utilización de la herramienta PICBASIC PRO para la programación de microcontroladores facilita en gran medida el diseño, aunque cabe considerar que si se obtiene, a través del compilador, el ensamblador equivalente y posteriormente el

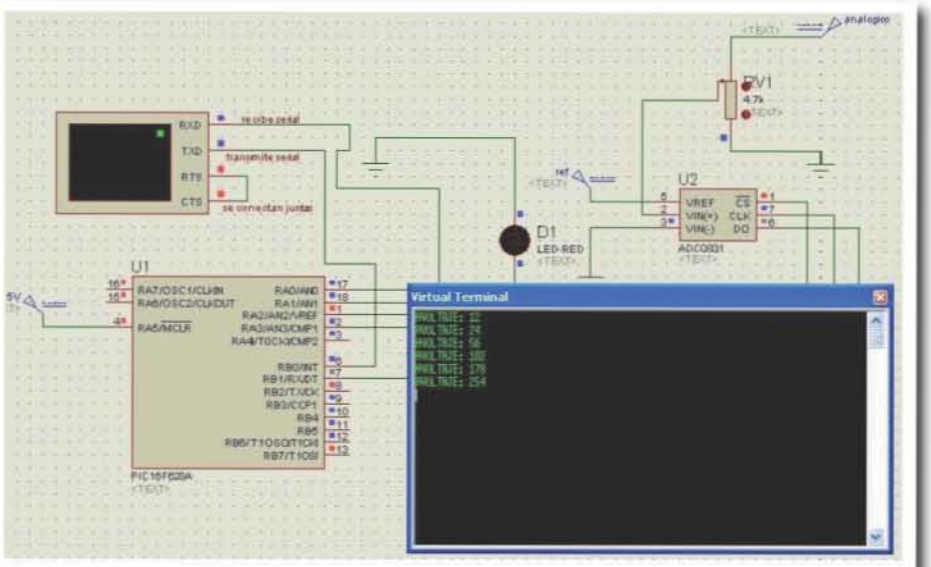

Figura 4. Aplicación que adquiere datos y los muestra.

archivo hexadecimal con el que se programará el microcontrolador, éste último implica un uso mayor de la memoria del dispositivo comparado con un diseño total en ensamblador. Aún así, el lenguaje de alto nivel resulta un recurso ampliamente socorrido por estudiantes y diseñadores profesionales.

Con respecto al proyecto que dio pie a este artículo, *"Sistemas de Control para Motores Supervisados por Computadoras de Bolsillo"*, cabe mencionar que la portabilidad y el poder de procesamiento de un ***PDA***

(Asistente Personal Digital) permiten concebir a estos dispositivos como sistemas altamente rentables para analizar datos en campo, sin la necesidad de otros aparatos de monitoreo y medición. La propuesta es programar bajo *Microsoft Windows CE* (actualmente *Windows Mobile*) para estudiar la factibilidad de la implementación de sistemas de control, idealmente de motores a CD, motores a pasos y servomotores, controlados por la Pocket PC vía el puerto serie y el protocolo RS-232.

Adicionalmente, se plantean pruebas utilizando la comunicación *RF* (radiofrecuencia) y la tendencia actual hacia *RFID* de identificación inalámbrica; ambas coinciden en el esquema de transmisión serial. La tecnología *RFID* reconoce códigos válidos a través de una señal de radiofrecuencia en un esquema de emisor - receptor tipo wireless (inalámbrico). En la actualidad existen innumerables aplicaciones que van desde sustituir un lector de código de barras, hasta descontar saldos en cuentas (tarjetas del Metrobús y boletos del metro, de la Ciudad de México).

Bibliografía

[1] Iovine, John; *PIC Microcontroller Project Book*, McGraw Hill, Segunda edición

[2] Jose M. Angulo Usategui; *Microcontroladores PIC: primera y segunda parte.* Tercera edición. Mc Graw Hill

[3] *Manual de PIC16F628/628A/648A.* Microchip, Inc.

[4] *Manual de PicBasic Pro Compiler.* MicrioEnginnering Labs. Inc.

[5] http://www.cidetec.ipn.mx/pofesores/jcrls/

[6] http://www.todopic.com.ar/pbp_sp/