COMENTARIO TECNICO

Técnicas de Comunicación Serie para Microcontroladores de la Flia. HC908 que no posean Módulo SCI

Por Hugo Lizzarraga / SyHDe – Soft y Hard Desarrollos
Email: info@syhde.com.ar
Adaptación: Ing. Daniel Di Lella / D.F.A.E For Motorola Products
Email: fae@electrocom.com.ar

INTRODUCCION:

Esta Aplicación describe el Software necesario para realizar una Interfase Serie Asincrónica sobre cualquier pin I/O estándar, para todos aquellos Microcontroladores de la Flia. HC908 que no posean un Módulo SCI.
Los requerimientos para que las rutinas funcionen correctamente son mínimos y, como es de esperarse poseen pequeños errores de Timming pero en la mayoría de los casos son inferiores al 1%, y en algunos casos no existen.
Cabe destacar que la rutina de Recepción Posee solamente control de Error de Encuadre (Frame) y no otros tipos de errores (Paridad, Ruido, etc).
La velocidad de Transmisión como la de Recepción son independientes, lo que nos permitiría usar la aplicación como un adaptador de baud rate si fuera necesario.

DESCRIPCION:

La presente Nota se basa en un Microcontrolador HC908 cualquiera, con un Cristal de **9.8304MHz**, lo que da una frecuencia de Bus de 2.4576MHz y un período T = 0.406901useg, con el que se calculan las rutinas y tablas de tiempo. La adaptación de las rutinas para algún eventual cambio en la frecuencia del Cristal no debería generar mayores inconvenientes, puesto que se incluye toda la información sobre el cálculo mismo y los diagramas de tiempo.
La aplicación está realizada como un archivo include: **SCI_TXRX.INC** para que su adaptación a cualquier programa de usuario sea mucho mas simple. El archivo include consta de tres rutinas principales:

saca_caracter - Rutina de transmisión sobre un pin TXD definido como salida, usa una tabla baud_tx para el seteo de la velocidad en la transmisión.

lee_caracter - Rutina de recepción sobre un pin RXD (Se recomienda el Bit 0 de cualquier puerto disponible) definido como entrada, y usa una tabla baud_rx para el seteo de la velocidad de recepción.

delay_7a - Rutina de tiempo, la cual es llamada por las rutinas de transmisión ó recepción.

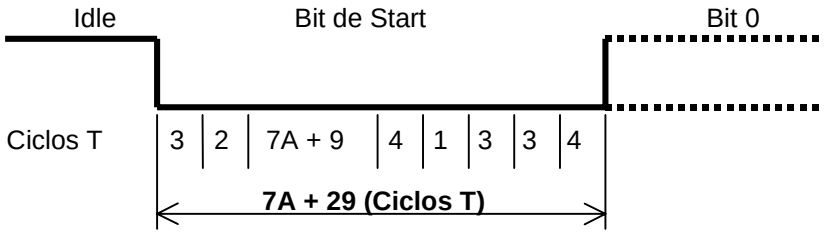
Minimizando los tiempos en el manejo de la comunicación obtenemos velocidades de hasta 57600 baudios con el cristal mencionado. Dejando abierta la posibilidad de aumentar la frecuencia de cristal para lograr velocidades mayores sin demasiado esfuerzo de cambio en las líneas de código.

TRANSMISION:

La rutina de Transmisión es la más simple puesto que una vez comenzada la transmisión y el bit de start haya arrancado, solo se trata de contar tiempo de acuerdo a la tabla baud_tx y cambiar el estado lógico del pin TXD ("0" ó "1") hasta contar los 8 bits del byte transmitido, luego se pone TXD = 1 (bit de STOP) y se espera ½ bit aproximadamente antes de buscar el siguiente byte a ser transmitido (suponiendo que ésa "búsqueda" nos consumirá el otro ½ bit restante antes de comenzar la transmisión del próximo byte).
El formato de transmisión es estándar NRZ n,8,1 y el orden de los bits es el siguiente:

bit START – bit 0 – bit 1 – bit 2 – bit 3 – bit 4 – bit 5 – bit 6 – bit 7 – bit STOP

El siguiente análisis de tiempo está basado en la rutina saca_caracter del archivo SCI_TXRX.INC y la unidad de medición de tiempo son los Ciclos T (0,406901 useg) de acuerdo a las condiciones especificadas anteriormente.



Este análisis nos lleva a calcular el valor de A más aproximado al ideal para cada baud rate, de lo cual sale la siguiente tabla:

Tabla 1.

BAUDIOS en TX	Ancho de bit Ideal (useg)	Valor de A hex – dec	Ciclos T (7A + 29)	Tiempo de bit (useg)	Diferencia al Ideal (useg)	Error (%)
57600	17,36	02 – 2	43T	17,49	0,13	0,74
38400	26,04	05 – 5	64T	26,04	0	0
19200	52,08	0E – 14	127T	51,67	-0,41	0,78
9600	104,2	20 – 32	253T	102,94	1,25	1,19
4800	208,3	45 – 69	512T	208,33	0,03	0,01
2400	416,7	8F – 143	1023T	416,25	-0,45	0,1

Una vez que el pin TXD subió a "1" Lógico comenzando el bit de STOP, no podemos esperar (7A + 9) Ciclos T para buscar el siguiente Byte a transmitir, como en los bits anteriores. Esto nos obliga a que luego de subir la línea TXD solo esperemos una fracción del tiempo necesario y ocupemos el resto en salir de la rutina é ir a buscar el siguiente byte.
Esto se hace menos problemático cuanto menor es la velocidad de transmisión.
El tiempo mínimo esperado para el bit de STOP es (7A+16) Ciclos T,
pero A = baud_tx/2.
Esto nos genera algunos inconvenientes para las siguientes velocidades:

57600 baudios: baud_tx = 2; A = 1; 23 ciclos T = 9.3587 useg; mientras que el bit Ideal = 17.36 useg. La diferencia es = 8 useg; o sea aprox 20 Ciclos T de Instrucciones para cargar el siguiente byte y volver a ingresar a la rutina de Tx. Si ahora tenemos en cuenta que este "reingreso" a la rutina saca_caracter, nos consume 18 Ciclos T para llegar al instante donde "bajamos" la linea TXD para comenzar el bit de START (previo seteo variables, etc.), vemos que es casi imposible cumplir exactamente con el bit de STOP.

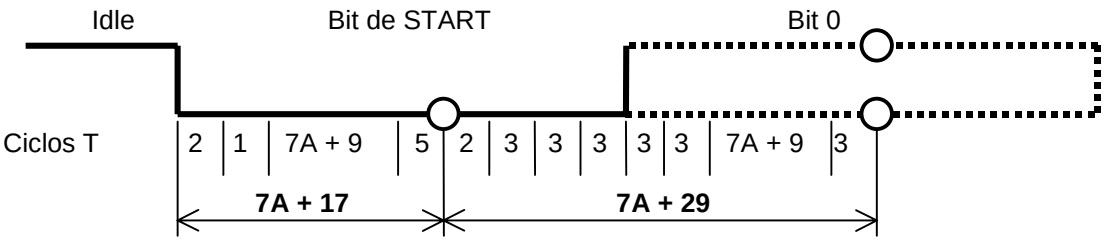
Para superar esta dificultad recomendamos que el receptor esté seteado a 2 bits de STOP.

Para el resto de las velocidades este problema no se presenta prácticamente. Por ejemplo:
38400 baudios: baud_tx = 5; A = 2; 30 Ciclos T = 12.2070 useg; mientras que el bit
Ideal = 26.04 useg. La diferencia es = 13.83 useg; o sea aprox. 34 Ciclos T.
Estamos un poco más cómodos con este tiempo.

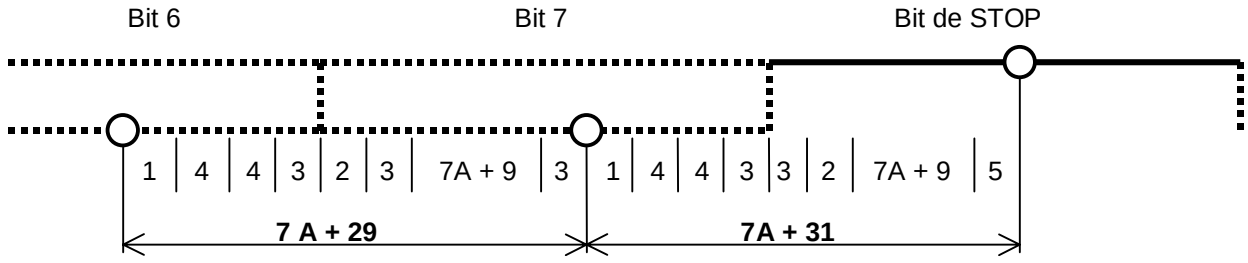
19200 baudios: baud_tx = 14; A = 7; 65 Ciclos T = 26.4485 useg; mientras que el bit
Ideal = 52.08 useg. La diferencia es = 25.64 useg; o sea aprox. 63 Ciclos T.

RECEPCION:

La rutina de Recepción funciona de la siguiente manera: Una vez que arrancó el bit de START, espero el tiempo correspondiente a ½ bit aproximadamente (7A + 17 Ciclos T donde A = baud_rx/2) y hago una lectura para asegurar el bit de START, a partir de aquí, muestreo el pin RXD a intervalos de tiempo correspondiente a un bit de ancho exactamente (7A + 29 Ciclos T donde A = baud_rx) para tomar el valor del bit en la mitad (donde la probabilidad de error es menor).
El siguiente análisis de tiempo esta basado en la rutina lee_caracter del archivo SCI_TXRX.INC y la unidad de medición de tiempo son los Ciclos T (0,406901 useg) de acuerdo a las condiciones especificadas anteriormente.



Una vez ingresados los 7 bits del byte recibido, debemos esperar un “1” lógico (bit de STOP), si esto no sucediera, tendríamos un error de encuadre. En tal caso la rutina de recepción anula el dato recibido y lo cambia por una bandera de error, que en este caso es el signo “$”. Esto se puede cambiar fácilmente para adaptar el soft a los requerimientos del usuario.



En este último análisis de tiempo podemos ver en detalle lo que sucede mientras se recibe el bit de STOP. Una vez que este haya sido leido, solamente tenemos ½ bit de ancho para guardar el byte recibido y reingresar en la rutina de recepción. Aquí ocurre lo mismo que en la transmisión, cuanto mayor es la velocidad, mayor inconveniente tenemos.
Una solución es que en el último muestreo acortemos el tiempo para leer el bit de STOP.

En el módulo 7A + 31 hagamos A = baud_rx – 2, lo que nos daría un poco mas de tiempo para guardar el dato recibido. Con esta solución prácticamente no tendríamos problema hasta 38400 baudios, pero a 57600 baudios esto es imposible por que el baud_rx para esa velocidad es = 2, y no podemos entrar a la rutina delay_7a con el Acc = 0. Para esta velocidad debemos cambiar la línea SUB #$02 por SUB #$01 ó DECA (Está indicado en el archivo fuente). Aun salvando este detalle, no podemos cumplir con los tiempos exactos para 57600 baudios, por lo que se recomienda que el transmisor esté seteado a 2 bits de STOP.

APLICACION:

La aplicación de estas rutinas es muy simple. Como primera medida debemos definir los pines TXD como salida, luego lo ponemos a “1” lógico y RXD como entrada, Este siempre debe ser el bit 0 de cualquier puerto (para facilitar el software). Luego definimos las variables carácter, cuenta_bit y auxiliar de 1 byte cada una, haciendo reserva de la memoria RAM para ellas. Por último debemos incluir el archivo SCI_TXRX.INC dentro de nuestro programa con la siguiente línea:

```
$INCLUDE  “SCI_TXRX.INC”    ;Inclusión del archivo de comunicación Serie
```

Ahora solo debemos cargar el dato a transmitir en la variable **carácter** y llamar a la rutina **saca_carácter** el dato será transmitido a la velocidad seteada.
Para la recepción, debemos llamar a la rutina **lee_caracter** y esperar a que salga, una vez que esto suceda, tenemos el dato recibido en la variable **carácter**.
Recordemos que si el dato recibido es “$” ($24) ha ocurrido un error de encuadre.

CONCLUSION:

El objetivo de esta Aplicación era realizar un Modulo SCI “virtual” en microcontroladores de muy bajo costo que no poseen uno “real”. Si tomamos una relación costo-beneficio creemos que el balance será ampliamente positivo aún después de “saltear” los pequeños inconvenientes para altas velocidades como se planteó anteriormente.

REFERENCIAS:

1) Nota de Aplicación AN1240 – Motorola Semiconductor by Scott George, CSIC MCU Product Engineering.

2) CPU08 Central Processor Unit Reference Manual – Motorola Semiconductor. Order Nro.CPU08RM/AD

3) Apuntes Propios sobre Microcontroladores Motorola – SyHDe, Soft y Hard Desarrollos.

APENDICE A:

```
*******************************************************************************
*               ARCHIVO INCLUDE - SCI_TXRX.INC                                *
* Rutinas para Comunicacion Serie Asincronica controladas por Soft.,aplicables  *
* a Microcontroladores de la Flia HC08 que no posean Modulo SCI.                *
* Caracteristicas:(p/fclk=9.8304Mhz, fbus=2.4576MHz, Ciclo T=0.406901useg)      *
*       Operacion en Half Duplex                                                *
*       Formato NRZ - N 8 1                                                     *
*       Velocidad Seleccionable 2400, 4800, 9600, 19200, 38400, 57600 baudios   *
*       Error en todos los casos menor al 1% (para 38400 baud, Error = 0)       *
*                                                                               *
*******************************************************************************

*******************************************************************************
* SACA_CARACTER - transmite la variable caracter por el pin TXD (Definido como  *
* Salida)                                                                       *
*                                                                               *
* Entrada       - caracter, variable donde viene el dato a transmitir           *
*                 Pin TXD definido como SALIDA y TXD = 1                        *
* Salida        - X, Acc, y caracter = Valor Indefinido                         *
* Stack Usado   - 2 Bytes                                                       *
* Var. Usadas   - caracter, almacena el dato a transmitir                       *
*                 auxiliar, almacena el valor de X                              *
* ROM Usada     - 36 Bytes                                                      *
*******************************************************************************
saca_caracter:
        stx     auxiliar        ;[3] guardo valor de X
        ldx     #9              ;[2] bits por caracter (incluido bit start)
        clc                     ;[1] borra carry para enviar bit de start
saca_bit_dato:
        bcc     saca_0          ;[3] si cy=0, salta para sacar 0
        bset    txd,ptb         ;[4] si no saca 1
        bra     otro_bit        ;[3]
saca_0:
        bclr    txd,ptb         ;[4]
        bra     otro_bit        ;[3]
otro_bit:
        lda     #baud_tx        ;[2]
        jsr     delay_7a        ;[7a+9]demora=7*31+9=226T
        ror     caracter        ;[4] busca siguiente bit (va al carry)
        decx                    ;[1]
        bne     saca_bit_dato   ;[3] 8T vuelvo a sacar el sig bit
saca_bit_stop:
        lda     #baud_tx        ;[2] Valor para TX a 19200 baudios
        lsra                    ;[1] A=baud_tx/2 divido en 2 a baud_tx
        bset    txd,ptb         ;[4]  __| sube linea para bit de STOP
        jsr     delay_7a        ;[7a+9] demora=16T=6,51useg
        ldx     auxiliar        ;[3]recupero valor de X
        rts                     ;[4]

;-------------------------------------------------------------------------------
; SELECCION DE VELOCIDAD DE TRANSMISION
;-------------------------------------------------------------------------------
baud_tx         equ     $05     ;Tabla de Seleccion de Baud Rate:
                                ; baud_tx       Velocidad
                                ;  $02     ->    57600 baudios
                                ;  $05     ->    38400 baudios
                                ;  $0e     ->    19200 baudios
                                ;  $20     ->     9600 baudios
                                ;  $45     ->     4800 baudios
                                ;  $8f     ->     2400 baudios
```

```
*****************************************************************************
* LEE_CARACTER - recive un dato por el pin RXD (siempre b0 de cualquier puerto) *
* y lo devuelve en la variable caracter                                       *
*                                                                             *
* Entrada       - El Pin RXD es definido como ENTRADA                         *
* Salida        - caracter, variable donde vuelve el dato recibido            *
*                 X, Acc Indefinidos                                          *
* Stack Usado   - 2 Bytes                                                     *
* Var. Usadas   - caracter, almacena el dato Recivido                         *
*                 cuenta_bit, variable temporal                               *
*                 auxiliar, almacena el valor de X                            *
* ROM Usada     - 61 Bytes                                                    *
*****************************************************************************
lee_caracter:
        stx     auxiliar                ;[3]guardo el valor de X
        lda     #$8                     ;[2]cant de bits del caracter
        sta     cuenta_bit              ;[3]
        clrx                            ;[1]
lee_bit_start:
        brset   rxd,ptb,*               ;[5]espera el bit de start
        lda     #baud_rx                ;[2] ($08)
        lsra                            ;[1] divido/2 el baud rate
        jsr     delay_7a                ;[7A+9]
        brclr   rxd,ptb,lee_b_dato      ;[5]=24T->9,76useg (mitad del pulso)
        bra     lee_bit_start           ;[3]


lee_b_dato:
        lda     #baud_rx                ;[2]\
        nsa                             ;[3] \
        nsa                             ;[3]  > Demora p/complementar 12 cicl.T
        nsa                             ;[3] /  q/faltan a la 1ra lectura (bit 0)
        nop                             ;[1]/
lee_bit_dato:
        lda     #baud_rx                ;[2]
        nop                             ;[1]
        nop                             ;[1]
        nop                             ;[1]
        jsr     delay_7a                ;[7a+9]
        lda     ptb                     ;[3] ->> Lee dato del b0,ptb (RXD)
        rora                            ;[1]
        ror     caracter                ;[4]
        dec     cuenta_bit              ;[4]
        bne     lee_bit_dato            ;[3]
lee_bit_stop:
        lda     #baud_rx                ;[3]
        sub     #$02                    ;[2] NOTA: p/57600 es (sub #$01)
        jsr     delay_7a                ;[7a+9]
```

```
        brset   rxd,ptb,rx_ok           ;[5] hasta aqui (7a+31)T A=1->38T=15,46us
        lda     #$24                    ; '$' -> caracter de error
        sta     caracter
rx_ok:
        ldx     auxiliar                ;[3] recupero el valor de X
        rts                             ;[4]

;-------------------------------------------------------------------------------
; SELECCION DE VELOCIDAD DE RECEPCION
;-------------------------------------------------------------------------------
baud_rx         equ     $05     ;Tabla de Seleccion de Baud Rate:
                                ; baud_rx       Velocidad
                                ;  $02     ->    57600 baudios
                                ;  $05     ->    38400 baudios
                                ;  $0e     ->    19200 baudios
                                ;  $20     ->     9600 baudios
                                ;  $45     ->     4800 baudios
                                ;  $8f     ->     2400 baudios

*********************************************************************************
* delay_7a - modulo de demora, sirve para calcular el tiempo en la velocidad de *
* comunicacion                                                                  *
* Entrada       - En el Acc. viene el modulo de demora, Dem.Tot.=(7*A+9) T      *
* Salida        - Acc = 0                                                       *
* Stack Usado   - 0 Bytes                                                       *
* Var. Usadas   - Ninguna                                                       *
* ROM Usada     - 7 Bytes                                                       *
*********************************************************************************
delay_7a:
        nop                     ;[1]
        nop                     ;[1]
        tsta                    ;[1]
        deca                    ;[1]
        bne     delay_7a        ;[3]= 7 cycles
        rts                     ;[4]=7a+4 cycles
*********************************************************************************
```

SyHDe – Soft y Hard Desarrollos
Email: info@syhde.com.ar
Web Site: www.syhde.com.ar