COMENTARIO TÉCNICO

Buceando en el HC908.....

Por Ing. Daniel Di Lella
Dedicated Field Application Engineer
www.edudevices.com.ar
dilella@arnet.com.ar

www.edudevices.com.ar

Como implementar un control remoto por infrarrojo en los microcontroladores HC908.

3ra. y última Parte.

Como habíamos visto en las dos entregas, nuestro sistema de control remoto por infrarrojo está formado por la unidad receptora, la cual fue descripta en detalle en los artículos anteriores, y por la unidad transmisora conocida vulgarmente como "el control remoto" del sistema. Esta unidad transmisora será la responsable de la emisión de la señal infrarroja codificada para que en forma telemática la unidad receptora decodifique la misma, y si corresponde, dispare una actividad o evento.

¿Que puntos son importantes a tener en cuenta a la hora de diseñar la unidad transmisora?..... Los puntos más importantes a tener en cuenta cuando hablamos de la unidad remota transmisora, son:

- Bajo consumo o nulo cuando la unidad no está operativa (sin emisión).
- Pequeño tamaño.
- Gran distancia de emisión efectiva (alcance).
- Bajo costo de implementación.
- Robustez general de la unidad, resistencia a las caídas, sin ajustes por golpes o por cambios climáticos, etc., etc.

La unidad aquí propuesta cumple con todas estas metas y además deja el campo libre para que el usuario la "mejore" con "chiches" de soft y/o hard que le otorguen mayor flexibilidad en sus aplicaciones.

Descripción de la unidad Transmisora Remota:

Nuestra unidad estará formada por dos grandes bloques, la "sección emisora" de señales infrarrojas y por el otro lado la "sección generadora" de señales.

Como se puede observar en la figura 1, nuestra sección emisora estará compuesta por 1, 2 o 3 LEDs emisores de infrarrojo, en configuración serie para mejorar el rendimiento de emisión de señales infrarrojas y lograr mayor eficiencia en el consumo de energía.

Recordando los artículos anteriores, en donde se describían las características principales del modulo receptor de infrarrojo IRM – 8601S de la firma Everlight, destacamos que el mismo basa su funcionamiento en la "sintonía" de señales infrarrojas con una frecuencia de 38Khz (frecuencia de mayor sensibilidad del modulo) y con un ciclo de actividad del 50% (cuadrada). Por ello, los LEDs emisores deberán modularse con dicha frecuencia y ciclo de actividad.
El modelo elegido para el proyecto es el popular "TSUS540" de la firma Vishay (ex Temic Semiconductors), pero el lector puede utilizar cualquier otro que cumpla con los parámetros principales del que hemos elegido aquí. Este diodo emisor de tecnología de Arseniuro de Galio, viene en formato 5 mm (T-1 3/4) y posee como características destacadas una alta eficiencia en la potencia irradiada versus la corriente de polarización del mismo, ángulo de emisión de 22º y longitud de onda de máxima emisión muy próxima a la necesaria.

Al diodo o conjunto de diodos, se lo manejará por medio de un simple transistor NPN de alto HFE y corriente de colector mayor a 300 mA. La corriente máxima que circulará por los mismos será del orden de los 200 mA. ya que el ciclo de actividad del 50 % de la señal, no permite una mayor estimulación de ellos.

La sección generadora, será la encargada de la generación de la señal de 38 Khz con el 50 % de ciclo de actividad y además de la codificación de la información a transmitir para que la misma llegue sin errores y en cualquier tipo de condiciones ambientales. La codificación responderá a lo implementado para la unidad receptora como se vió en los dos artículos anteriores y está basada en la codificación por ancho de pulso (PWM) de los "1" o "0" según un criterio similar al que utiliza la firma SONY para sus controles remotos.

El corazón de esta sección será un microcontrolador de la familia **HC908** como el **MC68HC908QT2CP** o similar, este MCU pertenece a la nueva camada de microcontroladores de 8 pines que incluyen oscilador interno con lo que se consigue un ahorro en pines disponibles para otras funciones y además del ahorro económico que ello involucra. Como el microcontrolador posee un conversor A/D podemos utilizar esta función para medir la tensión de la batería unos milisegundos previos a la transmisión, y con ello informar al usuario del estado de batería baja y evitar la transmisión de información posiblemente errónea.

Todo el sistema se alimentará de una pila alcalina miniatura de 12Vdc modelo "23 A", en el caso de la sección emisora trabajará con los 12Vdc en forma directa, mientras que en el microcontrolador se recurrirá al uso de un regulador 78L05 en formato TO-92 o similar para obtener los + 5v necesarios.

De esta forma, se cumple con uno de los puntos importantes a tener en cuenta que es el del bajo consumo o nulo, cuando no se utiliza la unidad remota, ya que si se observa el circuito de la **figura 1**, la alimentación al sistema solo se habilita al presionar el pulsador SW1, comenzando a funcionar a partir de ese momento.

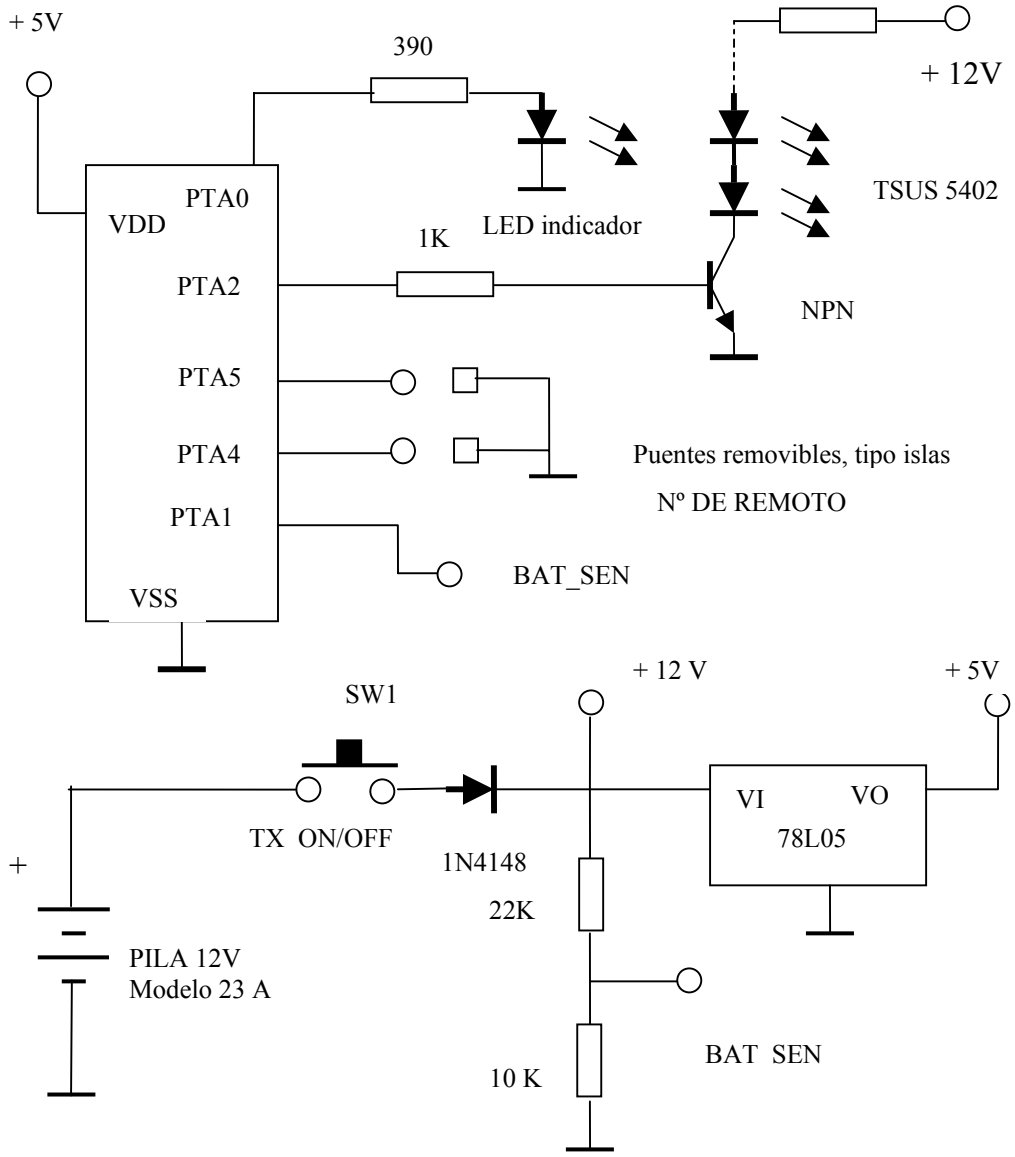

Figura 1 – Diagrama eléctrico de la unidad de Control Remoto por Infrarrojo

A continaución detalleremos el programa en assembler implementado en el MCU FLASH de 8 pines MC68HC908QT2CP:

```
*************************************************************************************************
*  TX_REMOTO -    SOFTWARE UNIDAD CONTROL REMOTO                                                 *
*                                                                                                *
*  EN ESTE EQUIPO SE UTILIZARA UN MCU MC68HC908QT2CP (DIP 8 PINES)                               *
*                                                                                                *
*  DEFINICIONES:                                                                                 *
*                                                                                                *
*  PUERTOS:                                                                                      *
*                                                                                                *
*  PORTA:                                                                                        *
*                                                                                                *
*  PTA0 --- OUT -- Control de LED indicador "TX ON"  / "Batería Baja"                            *
*                                                                                                *
*  PTA1 --- IN A/D -- Nivel de Batería                                                           *
*                                                                                                *
*  PTA2 --- OUT -- Control de ARRAY de LED infrarrojos (TSUS 5402)                               *
*                                                                                                *
*  PTA3 --- OUT -- NO SE USA en este proyecto !                                                  *
*                                                                                                *
*  PTA4 --- IN -- Número de Control Remoto (jumper x islas)                                      *
*                                                                                                *
*  PTA5 --- IN -- Número de Control Remoto (jumper x islas)                                      *
*                                                                                                *
*                                                                                                *
*  CONCIDERACIONES GRALES:                                                                       *
*                                                                                                *
*    Se usará el OSCILADOR INTERNO de los HC908Q                                                 *
*    Frecuencia de operacion --- Fbus = 3,200 Mhz.                                               *
*                                                                                                *
*                                                                                                *
*************************************************************************************************


*****************
* EQUATES   *
*****************


    base 10T              ; BASE DECIMAL POR DEFAULT

    $include 'qtqy_registers.inc' ; Este archivo contiene las definiciones
                                  ; con las direcciones de memoria
                                  ; los distintos registros del MC68HC908QT2


RAM          EQU  $0080    ; Comienzo de la RAM para el HC908QT2
RomStart     EQU  $F600    ; Comienzo de la ROM/FLASH para el programa
VectorStart  EQU  $FFDE    ; Comienzo tabla de vectores varios
CALI_OSC     EQU  $FFC0    ; Dirección donde se almacena de fabrica
                           ; valor calibración OSCILADOR INTERNO (NO BORRAR!)
T500msL      EQU  $A8      ; Valor hexa LOW base de tiempo del TIMER 500mseg
T500msH      EQU  $61      ; Valor hexa HIGH base de tiempo del TIMER 500ms
ADRESS       EQU  $08      ; Valor hexa GRUPO p/ control remoto (setear!!)
VNORMAL      EQU  $A0      ; Valor hexa V. normal BATERIA 12VDC (A/D 3,12V)
HEADER1      EQU  $1F      ; Valor hexa 1er HEADER a transmitir (1 1111)
HEADER2      EQU  $0F      ; Valor hexa 2do y 3er HEADER a transmitir (1111)
N_CERO       EQU  $17      ; Ancho del pulso "cero" en ciclos de RF (38KHZ)
N_UNO        EQU  $35      ; Ancho del pulso "uno" en ciclos de RF (38KHZ)
LED_I        EQU  0        ; Equate p/ PTA0 -- LED INDICADOR TX ON / BATERIA BAJA
LED_IR       EQU  2        ; Equate p/ PTA2 -- LED IR (ARRAY) INFRARROJO (TSUS 5402)
```

```
*****************************
* VARIABLES EN RAM *
*****************************

   ORG RAM                    ; Comienzo RAM para HC908QT2

TEMPA          RMB 1          ; M. temporal del ACC en RAM
TEMPXH         RMB 2          ; M. temporal de los reg. XH en RAM
AUX            RMB 1          ; Variable auxiliar multipropósito
NUremoto       RMB 1          ; Valor en RAM del numero de remoto
ADRESS1        RMB 1          ; Valor en RAM del ADRESS GROUP
VADCL          RMB 1          ; Sumatoria de valores A/D LOW
VADCH          RMB 1          ; Sumatoria de valores A/D HIGH
ADLOOP         RMB 1          ; Cdor. de N sumatorias val. A/D
VBAT           RMB 1          ; Valor convertido de BATERIA (40mSeg)
NU_TX          RMB 1          ; Cdor de STRINGS enviados
DATOS          RMB 1          ; Valor de datos recibidos x I.R.
HEADER         RMB 1          ; HEADER recibido por I.R.
NBIT           RMB 1          ; Numero de bits a transmitir
NLOOP          RMB 1          ; N de loops p/ un ancho de pulso
DATO           RMB 1          ; Valor en RAM del DATO a trans.
TI_OV          RMB 1          ; Flag indic. de TIME OUT (1,7SEG)

*********************************
* FLASH DE PROGRAMA *
*********************************

   ORG RomStart

********************
* Config. del MCU  *
********************

START   LDA #CALI_OSC          ;Procedimiento p/ ajustar OSCILADOR INTERNO
        STA OSCTRIM            ;a 3,2 Mhz con valor de fabrica en posición
                               ;FLASH en $FFC0 (NO HACER MASS ERASE !!)

        MOV #$01,CONFIG1       ;Reg. CONFIG1 - LVID=0,STOP=0,COPD=1
        MOV #$00,CONFIG2       ;Reg. CONFIG2 - IRQPUD,LVIT0=0,LVIT1=0
        NOP
        NOP                    ; NOPs de delay aseguran config.
        NOP
        NOP
        NOP
        MOV #$30,PTAPUE        ;R. PULL UP -- PTA4 / PTA5
        MOV #$30,PORTA         ;PORTA   - IN - PTA1,4,5 - OUT - PTA0,2,3,6,7
        MOV #$CD,DDRA          ;
        MOV #$30,PORTA         ;Fuerzo UNOS en PTA4 / PTA5
        NOP                    ;
        NOP
```

```
*************************
* HABILITO              *
* CONV. A/D             *
* DESHAB. IRQ EXT*
*************************

        MOV #$20,ADCLK      ;ADC Input Clock Reg. -- ADC clock / 2 (1mhz aprox.)
        MOV #$1F,ADSCR      ;ADC --- OFF -- A/D CH 1 (PTA1), OFF !!
        NOP
        MOV #$06,INTSCR     ;ACK1/IMASK1=1 ,MODE1=0 -- IRQ DESHAB.
        CLI                 ;I MASK=0 --- INT's habilitadas

*************************
* SETEO INICIAL DE *
* VARIABLES           *
*************************


        CLR AUX             ;Limpio variables en gral.
        CLR VBAT            ;
        CLR NUremoto        ;
        CLR ADRESS1         ;
        CLR VADCL           ;
        CLR VADCH           ;
        CLR ADLOOP          ;
        CLR DATOS           ;
        CLR HEADER          ;
        CLR NBIT            ;
        CLR NLOOP           ;
        CLR NU_TX           ;
        CLR TI_OV           ;
        CLR DATO            ;
```

```
************************************************************
* READ_DIP - RUTINA LECTURA DEL Nº REMOTO  *
*                                                          *
* Leo los jumpers / islas utilizando una                   *
* mascara de 1's y 0's para leer lo que me                 *
* interesa y luego haciendole una serie de                 *
* ROL'S y otras operaciones formo el                       *
* valor NUremoto que es de 8 bits                          *
*                                                          *
* jumpers / islas --- PTA5 / PTA4                          *
*                                                          *
* SACO --- NUremoto                                        *
*                                                          *
* ADRESS ---- DIRECCION DE REMOTO (GRUPO)  *
************************************************************


READ_DIP    MOV #ADRESS,ADRESS1    ;ADRESS1  --- #ADRESS (A RAM)
            LDA PORTA              ;Leo PORTA
             AND #$30              ;Mascara en bits 5 / 4 son los que
            CLC                     ;me interesan y los roto a b1/b0
            ROLA                   ;
            ROLA                   ;
            ROLA                   ;
            ROLA                   ;
            ROLA                   ;
            LSL ADRESS1             ;Luego desplazo a la IZQUIERDA el
            LSL ADRESS1            ;numero de GRUPO y se lo sumo
            LSL ADRESS1            ;Al ACC por una funcion OR
            LSL ADRESS1            ;
            ORA ADRESS1            ;
            STA NUremoto           ;ACC --  NUremoto (GRUPO + JUMPER / ISLA)
            BRA V_BATERIA          ;Voy a rutina medición V. BATERIA !


************************************************************
* V_BATERIA - RUTINA DE LECTURA V. BATERIA *
*          x A/D CH1 --  PTA1                              *
* SI VBAT ES >o= A VNORMAL-- SIGO ADELANTE *
* SINO - RUT. ERROR INDIC. X LED FLASHING      *
* VNORMAL=10 VDC en bateria - BAT_SEN=3,12V  *
* VNORMAL = $A0                                            *
************************************************************


V_BATERIA   BSET #LED_I,PORTA      ;ENCIENDO LED INDICADOR P/ CONSUMIR I
            CLR VADCL              ;Limpio variables p/ sumatoria
            CLR VADCH              ;de valores A/D de V de Linea
            CLR ADLOOP             ;
NEW_CONV1   MOV #$20,ADCLK      ;ADC Input Clock Reg. -- ADC clock / 2 (1mhz aprox.)
            MOV #$01,ADSCR     ;ADC --- ON -- A/D CH 1 (PTA1), NO INT'S, ONE CONV.
Coco_1?     BRCLR 7,ADSCR,Coco_1?   ;si coco=1 -- puedo leer ADR
            LDA ADR            ;sumo VADCL y lo guardo nuevamente
            ADD VADCL           ;
            STA VADCL           ;
            LDA #$00           ;SUMO con Carry el exceso de VADCL
            ADC VADCH           ;
            STA VADCH          ;Y lo guardo en VADCH
            JSR DLY1           ;Espero 1 mSeg entre muestra y muestra
            LDA ADLOOP         ;Tomé las 40 muestras ? (40 mS)
```

```
                CMP #39                 ;SI -- Procedo a promediarlas !!
                BEQ PROMED1             ;sino sigo tomando muestras...
                INC ADLOOP              ;
                BRA NEW_CONV1           ;

PROMED1     LDHX VADCL              ;DIVIDO LA SUMATORIA POR 40 para
                LDA VADCL               ;promediar el valor y  luego lo
                LDX #40                 ;GUARDO en VBAT !!
                DIV                     ;
                STA VBAT                ;

                LDA VBAT                ;VBAT > o = A VNORMAL ?
                CMP #VNORMAL            ;SI -- Sigo adelante !!!
                BHS SET_TIMER           ;Sino -- ERROR !! -- LED INDICADOR FLASHING!

VBAT_ERROR      BCLR #LED_I,PORTA       ;APAGO LED INDICADOR -- FLASHING
                JSR DLY250              ;DELAY DE 250MS
                BSET #LED_I,PORTA       ;ENCIENDO LED INDICADOR -- FLASHING
                JSR DLY250              ;DELAY DE 250MS
                BRA VBAT_ERROR          ;ME QUEDO AQUI FOREVER !!!


*****************************************************************
* SET_TIMER - Rutina de seteo del TIMER PATRON   *
*           que generará una INT x TOI/TOF                      *
*  cada 500 mSeg. APROX.                                        *
* FBus = 3,2 Mhz -- 312,5 nS de bus cicle                       *
* Prescaler = 64 -- 500mS / 20uS = 25000 cuentas                *
* MODH = $61 y MODL = $A8  -- HEX. 61A8                         *
*                                                               *
* T500msH = $61 y T500msL = $A8                                 *
*****************************************************************

SET_TIMER   MOV #$76,TSC            ;TSTOP,TRST,PS2/PS0,TOIE=1
                MOV #$00,TSC0           ;NO uso pines externos en el TIMER
                MOV #$00,TSC1           ;

                MOV #T500msH,TMODH      ;CARGO el Timer Module High
                MOV #T500msL,TMODL      ;CARGO el Timer Module Low ->$61A8
                LDA TSC                 ;Leo el TSC para borrar el TOF...
                BCLR 7,TSC              ;LIMPIO TOF en el TSC...
                BCLR 5,TSC              ;TSTOP=0 -- ARRANCO TIMER !!
                BCLR #LED_I,PORTA       ;APAGO LED INDICADOR -- ANTES DE TRANSMITIR



*****************************************************
* DELTA - RUTINA PPAL. DE TX POR I.R.     *
*                                                   *
* Se enviaran 3 strings completos durante           *
* el tiempo del TIMER (1,7 SEG APROX. )             *
* Para luego volver a empezar.....                  *
*                                                   *
* Portadora -> 38 KHZ -- Ton = Toff = 13uS          *
*                                                   *
* T. del "1" --- 1.4 mSeg.                          *
* T. del "0" --- 600 uSeg                           *
* T. de INTER - BIT --- 500 uSeg                    *
*                                                   *
*****************************************************
```

```
DELTA       LDA NU_TX            ;NU_TX=3? -- FIN DE LOS STRINGS !
            CMP #$03             ;sino sigo enviando STRINGS
            BEQ TX_END!          ;Voy a un LOOP indefinido ....
            BRA TRANS_IR         ;Voy a enviar STRINGS de I.R.


TX_END!     BCLR #LED_I,PORTA        ;APAGO LED INDICADOR -- FIN DEL ENVIO !!
            BCLR #LED_IR,PORTA       ;APAGO LED IR -- FIN DEL ENVIO !!
            BRSET 0,TI_OV,REINICIO   ;Espero en LOOP hasta TI_OV=1
            BRA TX_END!              ;

REINICIO    CLR NU_TX            ;Limpio cdores. de STRINGS y BITS
            CLR NBIT             ;
            BRA DELTA            ;Vuelvo a DELTA y todo comienza !


TRANS_IR    BSET #LED_I,PORTA      ;ENCIENDO LED INDICADOR -- NUEVO STRING !
            INC NU_TX              ;Cuento los STRINGS enviados
            LDA NU_TX              ;Es el 1er String ?
            CMP #$01               ;SI-- mando HEADER1 -- 1 1111
            BEQ HD_1               ;NO -- mando HEADER2 -- 1111
            MOV #HEADER2,HEADER    ;HEADER2 -- 1111 ($0F) 4 BITS VAL.
            MOV #$04,NBIT          ;NBIT=4 -- N. DE BITS A TRANS.
            ASL HEADER             ;1er desplazamiento p/ TRANS.
            BRA DESPLA_BIT         ;y luego sigo con los demas....

HD_1        MOV #HEADER1,HEADER    ;HEADER1 -- 1 1111 ($1F) 5 BITS VAL.
            MOV #$05,NBIT          ;NBIT=5 -- N. DE BITS A TRANS.
            BRA DESPLA_BIT         ;Voy a desplazar bits p/ trans.


DESPLA_BIT  ASL HEADER             ;Desplazo 3 lugares a la izq.
            ASL HEADER             ;el HEADER p/ colocarlo en posicion
            ASL HEADER             ;Al comenzar la TRANSMISION
ALFA        CLC                ;Carry = 0 -- envio de a un bit
            ROL HEADER             ;cont. del HEADER p/ TX "1" o "0"
            BCS TX_UNO             ;"1" -- 1.4mS , "0" -- 600 uS
TX_CERO     MOV #N_CERO,NLOOP      ;
LOOP_0      BSET #LED_IR,PORTA     ;LED IR -- ON -- I.R. ON !!
            BRN LOOP_0             ;Delay de 39 ciclos = 13 uS aprox.
            BRN LOOP_0             ;BRN = 3 CICLOS
            BRN LOOP_0             ;
            BRN LOOP_0             ;
            BRN LOOP_0             ;
            BRN LOOP_0             ;
            BRN LOOP_0             ;
            BRN LOOP_0             ;
            BRN LOOP_0             ;
            BRN LOOP_0             ;
            BRN LOOP_0             ;
            NOP                ;
            NOP                ;
            NOP                ;
            BCLR #LED_IR,PORTA     ;LED IR -- OFF -- I.R. OFF !!
            BRN LOOP_0             ;Delay de 39 ciclos = 13 uS aprox.
            BRN LOOP_0             ;
            BRN LOOP_0             ;
```

```
                BRN LOOP_0          ;
                BRN LOOP_0          ;
                BRN LOOP_0          ;
                BRN LOOP_0          ;
                BRN LOOP_0          ;
                BRN LOOP_0          ;
                BRN LOOP_0          ;
                NOP                 ;
                DBNZ NLOOP,LOOP_0     ;Si no es cero -- voy x prox. ciclo
                BRA DECRE_BIT         ;Voy al prox. bit a transmitir ...

TX_UNO          MOV #N_UNO,NLOOP      ;
LOOP_1          BSET #LED_IR,PORTA    ;LED IR -- ON -- I.R. ON !!
                BRN LOOP_1            ;Delay de 39 ciclos = 13 uS aprox.
                BRN LOOP_1          ;BRN = 3 CICLOS
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                NOP               ;
                NOP               ;
                NOP               ;
                BCLR #LED_IR,PORTA    ;LED IR -- OFF -- I.R. OFF !!
                BRN LOOP_1          ;Delay de 39 ciclos = 13 uS aprox.
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                BRN LOOP_1          ;
                NOP               ;
                DBNZ NLOOP,LOOP_1     ;Si no es cero -- voy x prox. ciclo
                BRA DECRE_BIT       ;Voy al prox. bit a transmitir ...


DECRE_BIT       DBNZ NBIT,INTBIT1     ;SI NBIT no es cero sigo TRANS.
                BRA INTBIT2         ;Si 0 - fin header y viene el dato

INTBIT1         LDA #01             ;ACC -- 1 para llegar a 500 uS
OUTLP23         LDHX #200           ;Loop interno de 500 uS ya que:
INNRLP23        AIX #-1             ;200 x 8 ciclos x 0,3125 uS = 500uS
                CPHX #0             ;
                BNE INNRLP23        ;
                DECA                ;ACC = 200,199,1...0
                BNE OUTLP23         ;1 x 500uS = 500uSeg
                JMP ALFA            ;Voy a buscar nuevo BIT !!

INTBIT2         LDA #01             ;ACC -- 1 para llegar a 1mS
OUTLP24         LDHX #200           ;Loop interno de 500 uS ya que:
INNRLP24        AIX #-1             ;200 x 8 ciclos x 0,3125 uS = 500uS
                CPHX #0             ;
                BNE INNRLP24        ;
```

```
                DECA            ;ACC = 200,199,1...0
                BNE OUTLP24         ;1 x 500 uS = 500uSeg
                BRA BETA            ;Voy a buscar el DATO a transmitir


****************************************
* BETA - SACO DATO POR I.R. *
****************************************


BETA            MOV #$08,NBIT       ;NBIT=8 -- N de Bits de "DATO" !
                MOV NUremoto,DATO   ;NUremoto -- DATO para enviar
GAMMA           CLC             ;CARRY = 0 -- Preparado p/ enviar
                ROL DATO            ;DATO x I.R.
                BCS TX_UNO2         ;"1" -- 1.4mS , "0" -- 600 uS
TX_CERO2        MOV #N_CERO,NLOOP   ;
LOOP_02         BSET #LED_IR,PORTA  ;LED IR -- ON -- I.R. ON !!
                BRN LOOP_02         ;Delay de 39 ciclos = 13 uS aprox.
                BRN LOOP_02         ;BRN = 3 CICLOS
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                NOP             ;
                NOP             ;
                NOP             ;
                BCLR #LED_IR,PORTA  ;LED IR -- OFF -- I.R. OFF !!
                BRN LOOP_02         ;Delay de 39 ciclos = 13 uS aprox.
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                BRN LOOP_02         ;
                NOP                 ;
                DBNZ NLOOP,LOOP_02  ;Si no es cero -- voy x prox. ciclo
                BRA DECRE_BIT2      ;Voy al prox. bit a transmitir ...

TX_UNO2         MOV #N_UNO,NLOOP    ;
LOOP_12         BSET #LED_IR,PORTA  ;LED IR -- ON -- I.R. ON !!
                BRN LOOP_12         ;Delay de 39 ciclos = 13 uS aprox.
                BRN LOOP_12         ;BRN = 3 CICLOS
                BRN LOOP_12         ;
                BRN LOOP_12         ;
                BRN LOOP_12         ;
                BRN LOOP_12         ;
                BRN LOOP_12         ;
                BRN LOOP_12         ;
                BRN LOOP_12         ;
                BRN LOOP_12         ;
                BRN LOOP_12         ;
                NOP             ;
```

```
                NOP             ;
                NOP             ;
                BCLR #LED_IR,PORTA      ;LED IR -- OFF -- I.R. OFF !!
                BRN LOOP_12             ;Delay de 39 ciclos = 13 uS aprox.
                BRN LOOP_12             ;
                BRN LOOP_12             ;
                BRN LOOP_12             ;
                BRN LOOP_12             ;
                BRN LOOP_12             ;
                BRN LOOP_12             ;
                BRN LOOP_12             ;
                BRN LOOP_12             ;
                BRN LOOP_12             ;
                NOP             ;
                DBNZ NLOOP,LOOP_12      ;Si no es cero -- voy x prox. ciclo
                BRA DECRE_BIT2          ;Voy al prox. bit a transmitir ...


DECRE_BIT2      DBNZ NBIT,INTBIT3       ;SI NBIT no es cero sigo TRANS.
                BRA INTBIT4             ;Si 0 -- fin del dato

INTBIT3         LDA #01                 ;ACC -- 1 para llegar a 500 uS
OUTLP25         LDHX #200               ;Loop interno de 500 uS ya que:
INNRLP25        AIX #-1                 ;200 x 8 ciclos x 0,3125 uS = 500uS
                CPHX #0                 ;
                BNE INNRLP25            ;
                DECA                    ;ACC = 200,199,1...0
                BNE OUTLP25             ;1 x 500uS = 500uSeg
                JMP GAMMA               ;Voy a buscar nuevo BIT !!

INTBIT4         LDA #04                 ;ACC -- 4 para llegar a 2mS
OUTLP26         LDHX #200               ;Loop interno de 500 uS ya que:
INNRLP26        AIX #-1                 ;200 x 8 ciclos x 0,3125 uS = 500uS
                CPHX #0                 ;
                BNE INNRLP26            ;
                DECA                    ;ACC = 200,199,1...0
                BNE OUTLP26             ;4 x 500 uS = 2 mSeg
                CLR TI_OV               ;Limpio Flag de Time Over...
                JMP DELTA               ;Voy a buscar un nuevo STRING ..



*****************************************************************
* TIMER_IRQ - SUB-RUTINA DE TIMER OVERFLOW  *
*                                                               *
* Se entrara x TOF / TOI cuando el tiempo                       *
* finalice --- time over = 500 mseg aprox.                      *
* Entonces limpio todo p/ volver a comenzar                     *
*****************************************************************


TIMER_IRQ       BRSET 7,TSC,TOF?        ;TOF=1 -- limpio TOF con cero
                BRA TIMER_IRQ
TOF?            BCLR 7,TSC              ;TOF --- 0 , listo p/ prox. IRQ

                BSET 0,TI_OV            ;TI_OV = 1 -- ACT. Flag TIME OVER !

                CLR NU_TX               ;Lpio variables p/ volver a empezar
                CLR NBIT                ;
                CLR HEADER              ;
```

```
                    CLR NLOOP           ;

                    RTI                 ;**** Vuelvo a PPAL. ******


*********************************
* DLY1                          *
* RUTINA DELAY 1 mS     *
* con FBUS= 3,2 MHZ     *
* T= 0,3125 uSeg            *
*********************************

DLY1        STA TEMPA           ;Salvo ACC en RAM
            STHX TEMPXH           ;Salvo X:H en RAM
            LDA #01             ;ACC -- 1 para llegar a 1mS
OUTLP21     LDHX #400             ;Loop interno de 1mS ya que:
INNRLP21    AIX #-1             ;400 x 8 ciclos x 0,3125 uS = 1mS
            CPHX #0               ;
            BNE INNRLP21            ;
            DECA                ;ACC = 1...0
            BNE OUTLP21             ;1 x 1 mS = 1 mSeg
            LDA TEMPA           ;RECOBRO EL ACC DE LA RAM
            LDHX TEMPXH           ;RECOBRO X:H DE LA RAM
            RTS                 ;RETORNO


*********************************
* DLY250                        *
* RUTINA DELAY 250 mS   *
* con FBUS= 3,2 MHZ         *
* T= 0,3125 uSeg            *
*********************************

DLY250      STA TEMPA           ;Salvo ACC en RAM
            STHX TEMPXH           ;Salvo X:H en RAM
            LDA #250            ;ACC -- 250 para llegar a 250 mS
OUTLP22     LDHX #400             ;Loop interno de 1mS ya que:
INNRLP22    AIX #-1             ;400 x 8 ciclos x 0,3125 uS = 1mS
            CPHX #0               ;
            BNE INNRLP22            ;
            DECA                ;ACC = 250,249, 1...0
            BNE OUTLP22             ;250 x 1 mS = 250 mSeg
            LDA TEMPA           ;RECOBRO EL ACC DE LA RAM
            LDHX TEMPXH           ;RECOBRO X:H DE LA RAM
            RTS                 ;RETORNO
```

```
*****************************************************************************
* INTERRUPT VECTORS TABLE -                                                 *
*                                                                           *
* RESET / SWI / IRQ1 / ADC / Y NO ASIGNADOS ---- START                      *
* KEYBOARD INTERRUPT VECTOR ---- START                                      *
* TIM1 OVERFLOW VECTOR ---- TIMER_IRQ                                       *
*                                                                           *
*****************************************************************************

    ORG  VectorStart

      dw  START       ; ADC Conversion Complete Vector
      dw  START       ; Keyboard Vector
      dw  START       ; (No Vector Assigned $FFE2-$FFE3)
      dw  START       ; (No Vector Assigned $FFE4-$FFE5)
      dw  START       ; (No Vector Assigned $FFE6-$FFE7)
      dw  START       ; (No Vector Assigned $FFE8-$FFE9)
      dw  START       ; (No Vector Assigned $FFEA-$FFEB)
      dw  START       ; (No Vector Assigned $FFEC-$FFED)
      dw  START       ; (No Vector Assigned $FFEE-$FFEF)
      dw  START       ; (No Vector Assigned $FFF0-$FFF1)
      dw  TIMER_IRQ    ; TIM1 Overflow Vector
      dw  START       ; TIM1 Channel 1 Vector
      dw  START       ; TIM1 Channel 0 Vector
      dw  START       ; (No Vector Assigned $FFF8-$FFF9)
      dw  START       ; ~IRQ1
      dw  START       ; SWI Vector
      dw  START       ; Reset Vector
```

..... Hasta la próxima!!!!

www.edudevices.com.ar