Prácticas y Aplicaciones con Microcontroladores PIC (III). Los PIC16F84 y PIC16C71.

Dr. Eugenio Martín Cuenca
Ing. José Mª Moreno Balboa

Facultad de Ciencias. Universidad de Granada.
E-mail:emartin@goliat.ugr.es

CIRCUITERÍA DE RESET

Los PICs 16C84, 16F84 y 16C71 solo disponen de un pin de ***reset***, la patita ***MCLR***. Incorporan internamente circuitería de *reset* que entra en funcionamiento de forma automática una vez se conecta la alimentación. En estos hay que distinguir varios tipos de reset:

- Por conexión de la alimentación V_{DD}. (*Power-on Reset* : ***POR***).
- Reset durante el funcionamiento normal al activar ***MCLR***.
- Reset durante modo de reposo ***SLEEP*** al activar ***MCLR***.
- Reset al sobrepasar el contador del "*Perro Guardián*" - ***WDT***.
- Detección de fallo en la alimentación (*Brown-Out*)

Si no se necesita circuitería de reset, la patita ***MCLR*** se conecta directamente a V_{DD}, por ejemplo cuando la alimentación alcanza una velocidad de crecimiento de la tensión a mayor velocidad que 0,05 V / ms.

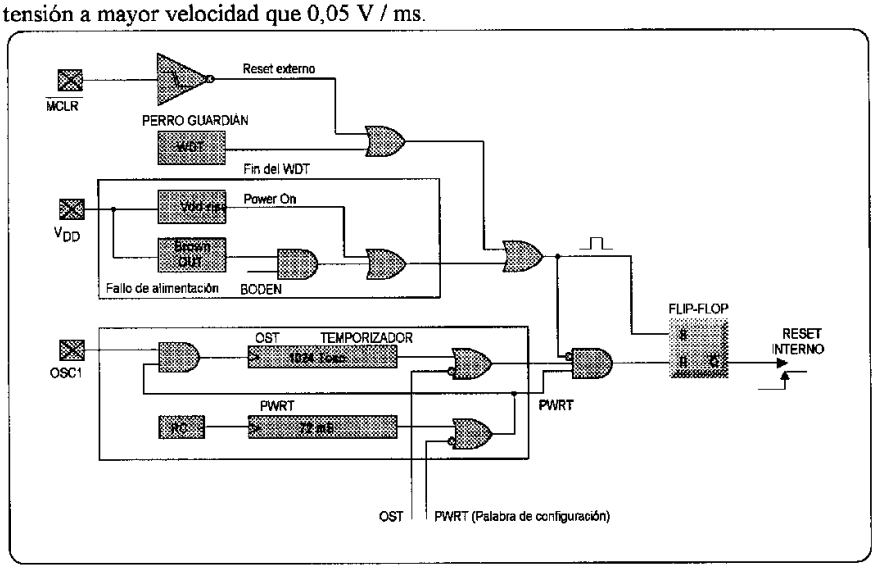

Figura 1.- *Esquema lógico del circuito que controla el flip-flop RS que guarda el estado del reset interno.*

R47

El temporizador *Power-up Timer* proporciona un retardo fijo de 72 ms durante el encendido. Este diseño mantiene el dispositivo en reset hasta que se estabiliza la alimentación.

La figura 1 muestra un diagrama de bloques simplificado de la circuiteria de reset incluida en el microcontrolador. La patita ***MCLR*** dispone de un filtro antiruido que ignora pulsos pequeños. Algunos registros no son afectados por ninguna condición de reset, su estado es desconocido durante el reset ***POR*** y no varían con cualquier otro reset. Muchos otros registros son afectados por el reset ***POR***, ***MCLR*** o ***WDT*** en funcionamiento normal o ***MCLR*** durante el modo reposo ***SLEEP***.

Power-on Reset (*POR*). Se genera un pulso de reset ***POR*** cuando se detecta la subida de VDD entre 1.2 y 1.7 V. Para disponer de esta ventaja es necesario llevar la patita MCLR a VDD directamente o a través de una resistencia. Esto elimina los componentes RC externos usualmente necesarios para el reset.

Power-up Timer (*PWRT*): Este temporizador proporciona un tiempo de 72 ms para ***POR***. Opera mediante un oscilador RC interno. El chip se mantiene en reset mientras ***PWRT*** esté activo. Esto permite que V_{DD} suba a un nivel aceptable. Este temporizador puede activarse o desactivarse durante la grabación del microcontrolador mediante el bit ***PWRTE*** (*Ver palabra de configuración*).

Oscilador Start-up Timer (*OST*): El oscilador Start-up proporciona un retardo de 1024 ciclos de reloj de OSC1 después de la finalización del retardo de ***PWRT***. Esto asegura que el cristal de cuarzo o el resonador haya arrancado y se ha estabilizado. Este temporizador solo actúa si se usan los modos XT, LP y HS y solo en el ***Reset Power-on*** o en wake-up en modo ***SLEEP***. Cuando VDD sube muy despacio, es posible que TPWRT time-out y TOST terminen antes de que VDD haya tomado su valor final. En este caso sería necesario una circuitería externa de reset power-on.

Time Out secuence y Power-Down status bits ($\overline{TO}$/$\overline{PD}$) : En Power-up la secuencia de time-out se invoca después de que ha finalizado ***POR***. Entonces se activa OST. El tiempo total de time-out varia dependiendo de la configuración del oscilador y del bit ***PWRTE***.

$\overline{POR}$	$\overline{BO}$	$\overline{TO}$	$\overline{PD}$	DESCRIPCIÓN
0	X	1	1	Power-On-Reset. Conexión de alimentación.
1	0	X	X	Reset por "Brown Out".
1	1	0	1	Reset del WDT en el modo normal.
1	1	0	0	Reset del WDT en el modo SLEEP.
1	1	1	1	Activación del MCLR en el modo normal.
1	1	1	0	Activación del MCLR en el modo SLEEP.

Tabla 1.- *Descripción del estado de los bits de STATUS en relacionados con el reset y su significado.*

Cuando se necesita control de reset externo, puede emplearse el esquema mostrado en la figura 2-A. También es posible conectar una circuitería de reset sensible a las variaciones de la tensión de alimentación, que sea capaz de activarse si esta

RAY

desciende por debajo de un determinado valor umbral (Brown-Out) figura 2-B. Reset por descenso de la tensión de alimentación.

	DIRECCIÓN	* Power-On-Reset	* MCLR en modo normal * MCLR en modo SLEEP * RESET provocado por el WDT * Brown-Out	* Final de SLEEP provocado por interrupción. * Final de SLEEP provocado por rebasamiento del WDT
W	---	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF	00h	----	----	----
TMR0	01h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCL	02h	0000 0000	0000 0000	PC + 1 (3)
STATUS	03h	0001 1xxx	000? ?uuu (4)	uuu? ?uuu (4)
FSR	04h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA	05h	---x xxxx	---u uuuu	---u uuuu
PORTB	06h	xxxx xxxx	uuuu uuuu	uuuu uuuu
EEDATA	08h	xxxx xxxx	uuuu uuuu	uuuu uuuu
PCLATH	0Ah	---0 0000	---0 0000	---u uuuu
INTCON	0Bh	0000 000x	0000 000x	uuuu uuuu (2)
INDF	80h	---- ----	---- ----	---- ----
OPTION	81h	1111 1111	1111 1111	uuuu uuuu
TRISA	85h	---1 1111	---1 1111	---u uuuu
TRISB	86h	1111 1111	1111 1111	uuuu uuuu
EEADR	09h	xxxx xxxx	uuuu uuuu	uuuu uuuu
EECON1	88h	---0 x000	---0 q000	---0 uuuu
EECON2	89h	---- ----	---- ----	---- ----

Tabla 2.- *Condición en reset de algunos registros.*

Reset en Brown-Out : Esta es una condición donde la alimentación del dispositivo VDD toma un valor inferior al mínimo, pero sin llegar a cero y luego se recupera. El microcontrolador entra en reset en ese momento. Para que este se produzca es necesario añadir una circuitería externa de reset por descenso de la alimentación como la presentada en la figura 2-B.

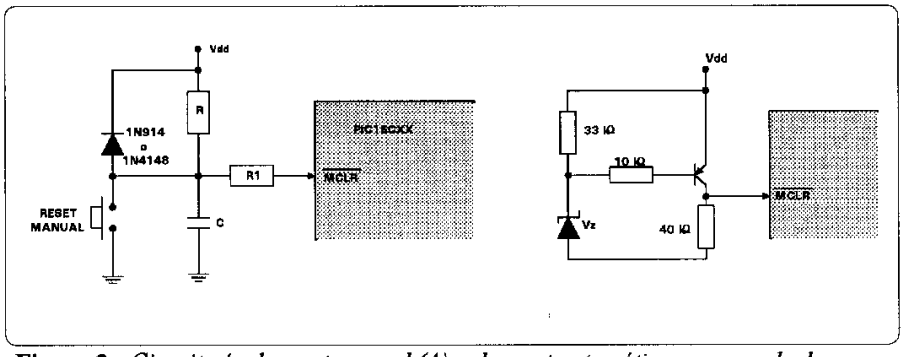

Figura 2.- *Circuitería de reset manual (A) y de reset automático en caso de descenso de la tensión de alimentación (B).*

PALABRA DE CONFIGURACIÓN

Se encuentra en la posición $2007 de la *memoria de programa*, dirección a la que únicamente puede accederse durante la grabación del microcontrolador. Al ser una posición de la memoria de programa se dispone por tanto de 14 bits cuya función es la siguiente:

Bits 4-13 CP ***Code Protection***	**1 : Protección de código off** **0 : Toda la memoria con código protegido**
Bit 3 $\overline{\text{PWRTE}}$ ***Power-up Timer***	**1 : Power-up timer desactivado** **0 : Power-up timer activado**
Bit 2 WDTE ***Watchdog Timer Enable***	**1 : WDT "*Perro Guardián* " activado** **0 : WDT "*Perro Guardián* " desactivado**
Bits 1 - 0 FOCS1 - FOSC0 ***Oscillator Selection***	**11 : Oscilador tipo RC** **10 : Oscilador tipo HS** **01 : Oscilador tipo XT** **00 : Oscilador tipo LP**

13													0
CP	CP	CP	CP	CP	CP	CP	CP	CP	CP	$\overline{\text{PWRTE}}$	WDTE	FOSC1	FOSC0

La palabra de configuración anterior es válida para el PIC16F84, pero en el caso del PIC16C84 y el PIC16C71 los bits ***$\overline{\text{PWRTE}}$*** están invertidos. Esta es una de las escasas diferencias entre los diferentes modelos. Desde el punto de vista del usuario, no es de gran importancia, pues esto se ha tenido en cuenta en el diseño del programa **PIC84/71.EXE** que controla el grabador (KIT que puede conseguirse a través de la empresa granadina **I.D.E.M.**, de la que damos la dirección al final del capítulo). Solo es necesario seleccionar el modelo de PIC a grabar y si solo cuando se realice protección de código.

RAD

Practicas (III)

MÓDULO GENERAL DE APRENDIZAJE

Con el programador **GCHIPS Labs PIC84/71**, el ***Módulo Básico*** del mes pasado, y este ***General de Aprendizaje*** que presentamos, puede realizar el 90 % de las practicas incluidas mi libro indicado en la bibliografía. Estos dos módulos se conectan enfrentados, ya que el Módulo Básico dispone de un conector macho y el módulo General de Aprendizaje de un conector hembra.

En algunos casos como puede ser el uso de interrupciones deberá ajustar los diferentes jumpers. En la configuración por defecto, las patillas de la Puerta A están conectadas a cinco pulsadores (A0 - A4) y las patillas de la Puerta B están conectadas a una barra de diodos electroluminiscentes y a 3 presentadores de 7 segmentos.

Si se desean realizar experimentos de interrupción, el jumper J3 permite llevar la línea RB0/INT al conector CN2 donde queda disponible. Para el uso del TMR0 a través de la entrada de la patita A4, el jumper J8 lleva esta línea al conector CN1.

Por último, los jumpers J4-J8 pueden llevar cualquiera de las líneas RA0-RA4 al conector CN1, lo que permite el empleo de estas líneas en funciones de entrada y salida, que en el caso de usar un PIC16C71 pueden programarse indistintamente como entradas A/D en las líneas A0-A3. Este es un empleo diferentes a las de su funcionamiento como pulsadores A0-A4.

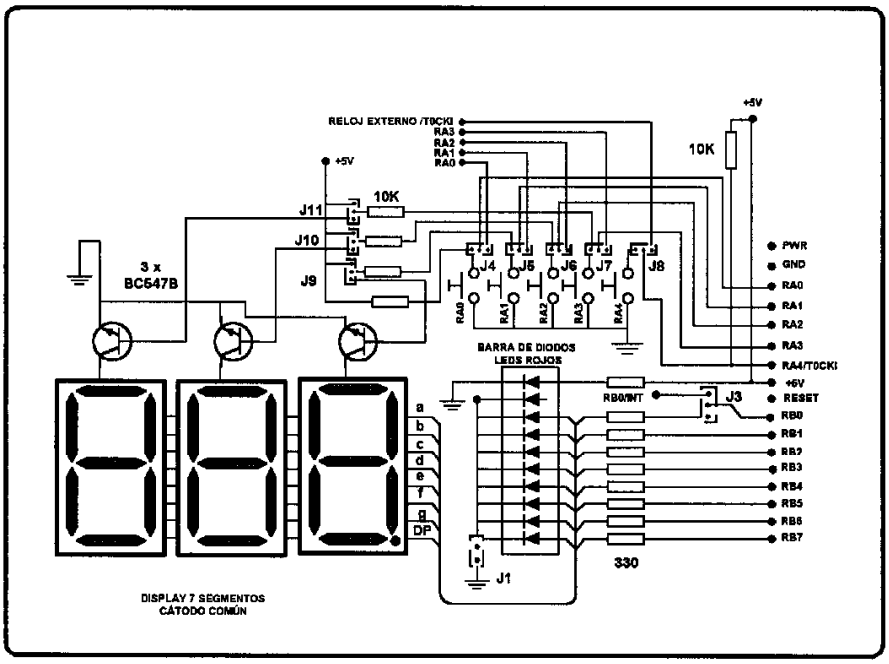

Figura 3.- *Diagrama eléctrico del Módulo General de Aprendizaje.*

Para emplear los presentadores de 7 segmentos, se debe cambiar de posición los jumpers (puentes) J9 (A1), J10 (A2) y J11 (A3) que gobiernan el encendido de cada uno de estos displays a través de transistores BC547B. Conviene en este caso desconectar la barra de diodos LEDs retirando el jumper J1.

La línea de alimentación se ha conectado a uno de los diodos LEDs de la barra cuya función es indicar la presencia de suministro eléctrico.

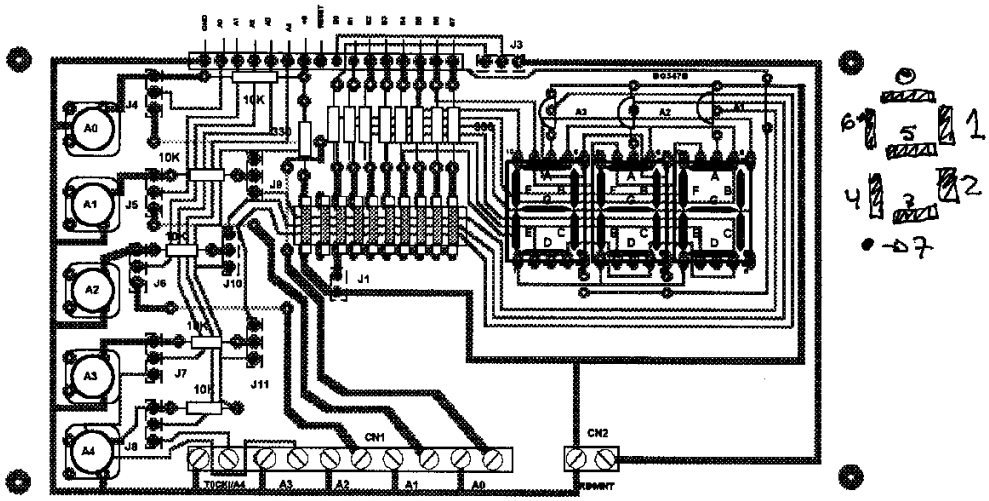

Figura 4.- *Disposición de componentes del Módulo General de Aprendizaje.*

Así por lo tanto, en este momento se ha suministrado al lector material suficiente para poder realizar sus diseños en la mayoría de los casos.

Se ha separado exprofeso la línea A4 al tener la característica adicional de ser la entrada de contador del Timer0. Con respecto a las líneas del Puerto B se han llevado a una barra de diodos LEDs y a 3 presentadores (displays) de 7 segmentos. Esto permite realizar las practicas de salidas de información.

EXPERIENCIA (2)

La segunda experiencia consiste en la escritura de un programa para el aprendizaje usando el ***Módulo Básico*** y el ***Módulo General de Aprendizaje***. Conecte los dos módulos tal y como se ve en la fotografía. Realizaremos la experiencia a título de ejemplo, aunque antes de seguir adelante debemos recordar que el programa CONTAR.ASM del capítulo anterior puede ejecutarse aquí perfectamente.

RAC5

```
;*****************************************************************************
;Programa: REDE001.ASM                    Fecha:  8/11/1997                  *
;*****************************************************************************
;    Genera un número para unos dados dobles                                 *
;    Interruptor en el BIT 0 del puerto A genera un número nuevo.            *
;                                                                            *
;                                                                            *
; Revisión: 1.0              Programa para el PIC(16C84/16F84)               *
; Velocidad de reloj: 4Mhz       Reloj instrucción: 1us                      *
;                                                                            *
; Reloj: tipo XT             Perro Guardián: Off                             *
;*****************************************************************************
;*****************************************************************************
;Definiciones                                                                *
;*****************************************************************************
;Direcciones de los registros de uso especifico del PIC 16C84
INDF        EQU   00h     ;Contenido de la dirección apuntada por FSR
FSR         EQU   04h     ;Dirección base del direccionamiento indirecto
PORTA       EQU   05h     ;Dirección del puerto A
PORTB       EQU   06h     ;Dirección del puerto B
STATUS      EQU   03h     ;Dirección del registro de estado
TRISA       EQU   05h     ;Dirección del registro de estado para A
TRISB       EQU   06h     ;Dirección del registro de estado para B
PCL         EQU   03h     ;Dirección del contador de programa bajo
;bits del registro STATUS
C           EQU   00h     ;Bit de acarreo
Z           EQU   02h     ;Bit de cero
RP0         EQU   05h     ;Selecciona el segundo banco de registros
;Definiciones utilizadas en el programa
NUMEMAX     EQU   067h    ;Número máximo + 1 del rango de salida(BCD)

;*****************************************************************************
;Posiciones de memoria utilizadas por el programa                            *
;*****************************************************************************
            ORG   0Ch
Contador1   res   1       ;Utilizada por la rutina de retardo
Contador2   res   1       ;Utilizada por la rutina de retardo
Contador3   res   1       ;Utilizada por la rutina de retardo
Temp0       res   1       ;Para almacenamiento temporal
Temp1       res   1       ;Para almacenamiento temporal
Semilla     res   1       ;Para la generación del número aleatorio
Numero      res   1       ;Guarda el número a mostrar

;*****************************************************************************
;Primera instrucción tras salir del estado de reset
            ORG   00h
            GOTO  INICIO
;*****************************************************************************
;Código correspondiente a los procedimientos utilizados
;
            ORG   05h
;*****************************************************************************
;Pasa de un número de BCD -> al código para el display.                      *
;entrada:  W -> contiene el número a presentar.                              *
;salida:   W -> código para el display                                       *
;variables utilizadas:                                                       *
;*****************************************************************************
PasaBCDDi   ADDWF  PCL,1
            RETLW  0feh       ;Código del 0
            RETLW  020h       ;Código del 1
```


RAG

```
                RETLW   03Dh        ;Código del 2
                RETLW   079h        ;Código del 3
                RETLW   022h        ;Código del 4
                RETLW   0B2h        ;Código del 5
                RETLW   01Fh        ;Código del 6
                RETLW   070h        ;Código del 7
                RETLW   0feh        ;Código del 8
                RETLW   072h        ;Código del 9
;*****************************************************************************************
;
;Coloca en pantalla los dos dígitos del número aleatorio.                               *
;entrada:  W -> contiene el número a presentar.                                         *
;          4 bits el número más  significativo                                          *
;          4 bits el número menos significativo                                         *
;salida:   nada                                                                         *
;variables utilizadas: Temp0                                                            *
;*****************************************************************************************
Pinta       MOVWF   Temp0       ;Guarda el número de entrada en Temp0
            ANDLW   0fh         ;Nos quedamos con los 4bits menos significativos
            CALL    PasaBCDDi   ;Pone en el código en el puerto B
            BSF     PORTA,1     ;Activa el dígito menos significativo
            NOP                 ;Producimos un retardo de 3us en el que
            NOP                 ;se muestra el dígito
            NOP                 ;
            BCF     PORTA,1     ;Desactiva el dígito menos significativo
            SWAPF   Temp0,0     ;Intercambia los 4 bits inferiores por los superiores
            ANDLW   0fh         ;nos quedamos con los 4bits más significativos
            CALL    PasaBCDDi   ;Pone en el código en el puerto B
            BSF     PORTA,2     ;Activa el dígito central
            NOP                 ;Producimos un retardo de 3us en el que
            NOP                 ;se muestra el dígito
            NOP                 ;
            BCF     PORTA,2     ;Desactiva el dígito central
            RETURN              ;Salimos de la subrutina
;*****************************************************************************************
;Incrementa Semilla una unidad.                                                         *
;entrada: Nada                                                                          *
;salida:  En Semilla se encuentra un número del 1 al MUNEMAX-1 (En BCD)                 *
;variables utilizadas: Semilla                                                          *
;*****************************************************************************************
IncSemilla  INCF    Semilla,1   ;Incrementa en una unidad Semilla
            MOVLW   0fh         ;Carga en W 0fh
            ANDWF   Semilla,0   ;Nos quedamos con los 4bits bajos
            SUBLW   0Ah         ;Vemos si es un número de 0 a 10
            BTFSC   STATUS,Z    ;Si es 10 se pone acero los 4bits bajos
            GOTO    IncSemiAc   ;y se incrementan los 4bits altos
            GOTO    IncSemiTo   ;de lo contrario vemos si ha llegado al número máximo
IncSemiAc   MOVLW   010h        ;Carga 010h en W
            ADDWF   Semilla,1   ;Suma a los 4bits altos de Semilla 1
            MOVLW   0F0h        ;Carga 010h en W
            ANDWF   Semilla,1   ;Ponemos los 4bits bajos a cero
IncSemiTo     MOVF  Semilla,0   ;Carga W con el valor de Semilla
            SUBLW   NUMEMAX     ;Resta W el número máximo
            BTFSS   STATUS,Z    ;Testamos el bit de cero
            RETURN              ;Si no es máximo sale de la subrutina (Z=0)
            MOVLW   01h         ;Si es el máximo Semilla toma el valor 1
            MOVWF   Semilla
            RETURN              ;Salimos de la subrutina
```


RAG

```
;*****************************************************************************************
;De un número pasado en W sale un número de 1 a 6 en W                                  *
;entrada: W    -> un número BCD del 0 al 9                                              *
;salida:  Temp0 -> un número de 1 a 6 generado a partir del número de entrada           *
;variables utilizadas: nada                                                             *
;*****************************************************************************************
Numero16    MOVWF     Temp0       ;Almacenamos el número en Temp0
            MOVLW     06h         ;Cargamos W con el número Máximo
NumeroLazo  SUBWF     Temp0,1     ;Restamos a Temp0 el contenido de W
            BTFSC     STATUS,C    ;Seguimos restando a Temp0 W, hasta
            GOTO      NumeroLazo  ;que Temp0 tenga un número negativo
            ADDWF     Temp0,1     ;Suma a Temp0 el número máximo Temp0
                                  ; número del 0 al 5
            INCF      Temp0,1     ;Temp0 pasa a ser un número del 1 al 6
            RETURN                ;Sale de la subrutina
;*****************************************************************************************
;Produce un retardo,presenta en pantalla una sucesión de números y                      *
;varia el valor de Semilla.                                                             *
;entrada          nada                                                                  *
;salida           nada                                                                  *
;variables utilizadas: Contador1,Contador2,Contador3                                    *
;*****************************************************************************************
EstadoEsta  MOVLW     0A0h        ;Carga en W el valor 0Ah
            MOVWF     Contador1   ;Carga contador1 con W
            MOVLW     030H        ;Carga en W el valor 030h
            MOVWF     Contador2   ;Carga contador2 con 030h
            MOVLW     01h         ;Carga en W wl valor 01h
            MOVWF     Contador3   ;Carga Contador3 con 01h
EsLazo1     MOVF      Semilla,0   ;Carga W con el contenido de Semilla
            CALL      Pinta       ;Pinta el valor en los displays
            DECFSZ    Contador1,1 ;Decrementa Contador1 y salta si es cero
            GOTO      EsLazo1     ;Cerramos el primer bucle
            CALL      IncSemilla  ;Incrementa en uno Semilla
            DECFSZ    Contador2,1 ;Decrementa Contador2 y salta si es cero
            GOTO      EsLazo1     ;Cerramos el segundo bucle
            CALL      IncSemilla  ;Incrementa en uno Semilla
            DECFSZ    Contador3,1 ;Decrementa Contador3 y salta si es cero
            GOTO      EsLazo1     ;Cerramos el tercer bucle
            RETURN                ;Salimos de la subrutina
;*****************************************************************************************
;Código correspondiente al programa principal
;
INICIO      BSF       STATUS,RP0  ;Seleccionamos el segundo banco
            CLRF      TRISB       ;El puerto B como salida
            MOVLW     0f1h        ;Carga en W el valor 255
            MOVWF     TRISA       ;Selecciona al puerto A :
                                  ;bit 0    -> entrada
                                  ;bit 1,2,3 -> salida
            BCF       STATUS,RP0  ;Seleccionamos el primer banco
            MOVLW     01h         ;Carga en W el número 01 en BCD
            MOVWF     Semilla     ;Carga Semilla con el valor W
            MOVLW     00h         ;Carga en W el número 00 en BCD
            MOVWF     Numero      ;Carga Numero con el valor de W
                                  ;El valor 00 de salida indica que el
                                  ;juego no ha empezado
NumeroNuevo BTFSC     PORTA,0     ;Si bit 0 de PORTA es 1 genera nuevo número
            GOTO      Refresca    ;De lo contrario pone en pantalla el antiguo
            CALL      EstadoEsta  ;Saca variaciones de números en pantalla
                                  ;y varia el valor para la semilla
NumeroMalo  CALL      IncSemilla  ;Actualiza el valor de la semilla
```


RAO