:: Microcontroladores PIC - Parte IV - Trabajando con Display's y Tablas.

Indice General:

Saludos para todos...!!!

R-Luis

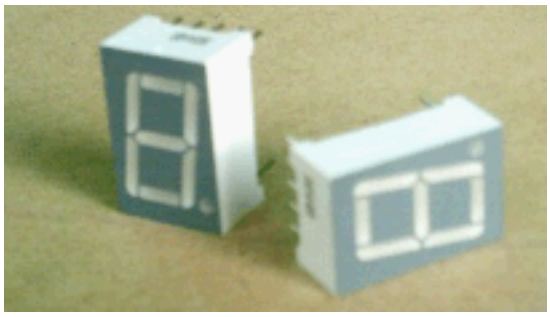
:: Microcontroladores PIC - Parte IV - Introducción

Voy a tratar de no hacerlo muy extenso, aunque me va a costar un poco, pensaba hacer algo con LCD, pero aquí en Jujuy, te sacan un ojo y te rompen el otro por un pequeñín de esos, asi es que lo haré con un simple Display, por otro lado, si alguien ya trabajó con los LCD, sería bueno que manden un tutorial al respecto, así lo incorporamos y ya estaríamos casi completitos, les parece...???. Ok, Comencemos...

Si bien todas las tareas que realizamos, las podemos ver reflejadas en un simple LED, hay ocasiones, en que es necesario ver un número o una letra que nos brinde información de aquello que nuestro micro está haciendo, o simplemente que nos muestre la hora (podríamos hacer un reloj...!!!), o que le muestre un mensaje a nuestra "FIEL" Amada (Te casarías conmigo...???), jaja, bueno, lo último está demás... :oD

Pues para eso están los LCD o los Displays

Para quienes no conozcan lo que es un Display, aquí tienen una imagen



Básicamente un Display es una colección de LEDs ubicados de forma estratégica, y como todo LED, obviamente, dispone de un Cátodo y un Ánodo, el tema es que como son varios LED's, se los agrupa uniendo sus cátodos en cuyo caso será de CÁTODO COMUN, o bien agrupando sus ánodos, resultando un Dislplay de ANODO COMUN, por otro lado estos LED's pueden ser fabricados en forma de Puntos o Segmentos, tal es así que se encuentran Displays de 7 segmentos, como los de la imagen (que son los más comunes de todos),

En fin, nosotros tabajaremos con un Display de CÁTODO COMÚN y de 7 segmentos, más el punto (por supuesto)

Si ya están preparados para comenzar, aquí vamos...

:: Microcontroladores PIC - Parte IV - Capitulo 1

Para comenzar, les contaré lo que haremos...

Vamos a hacer un programa que lea la cantidad de veces que se activa un pulsador y muestre el resultado correspondiente. Para hacerlo, tenemos dos posibilidades, una de ellas es hacerlo en forma directa, es decir conectar el puerto B del micro a los pines del Display, y luego encender cada uno de los segmentos del Display para visualizar el valor correspondiente.

La otra posibilidad es utilizar un decodificador BCD como el 74LS47 o el 74LS249, o el CD4511 que es el que yo utilizaré.

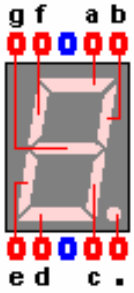
Estos integrados disponen de 4 entradas correspondientes a un código binario, y 7 salidas que se conectan a un Display para mostrar el valor en decimal, o en hexadecimal, según el caso, el nuestro sólo lo hará en decimal.

Yo trabajaré de las dos formas, con y sin decodificador, así tienen una idea de como trabajar con ellos...

Trabajando con un decodificador BCD

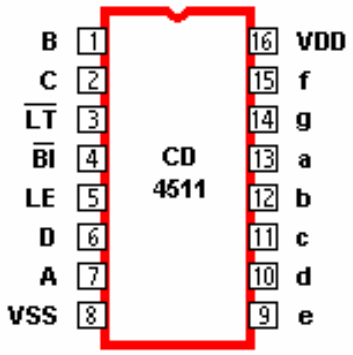
Primero veamos todos los componentes que vamos a utilizar

El primero de ellos, es un Display de 7 segmentos de cátodo comun, por ser de cátodo común, es obvio pensar que las señales que deberá recibir este Display para iluminar sus segmentos, deben ser positivas, aquí tienen una imagen del display y sus pines...



Este Display esta compuesto por 10 pines, de los cuales 7 corresponden al ánodo de cada segmento (nombrados como a, b, c, d, e, f y g), uno para el punto (.), y finalmente 2 que corresponden al cátodo, los cuales están pintados de azul, aquí hay que aclarar algo, estos dos terminales son comunes, así que da lo mismo que conectes cualquiera de ellos o los dos.

El segundo componente importante aquí es el Decodificador, y yo voy a trabajar con el CD4511.



Lo importante de este integrado, es que posee 4 pines de entrada y 7 de salida, mas unos cuantos de configuración. El hecho es que, los 4 pines de entrada (**A, B, C** y **D**) serán los que reciban el código en binario de la cantidad de veces que se activó el pulsador (dato enviado por el micro). Una vez recibido el dato, el integrado se hará cargo de decodificarlo y enviarlo por los pines de salida (**a, b, c, d, e, f** y **g**) para mostrarlo en el display, interesante no...!!!

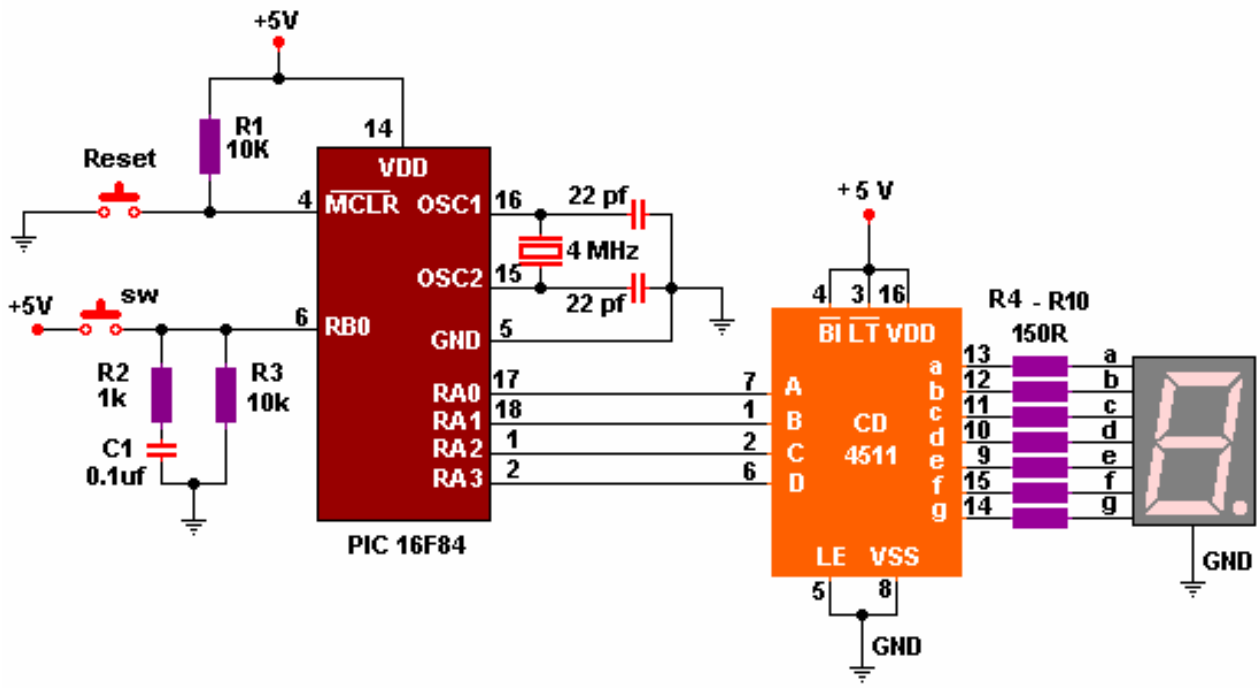
Lo que nos falta saber, es que dato deberé enviar al decodificador para que este muestreeeee... el cero por ejemplo, para esto no hay nada mejor que ver su tabla de verdad, y aquí está...

Entradas				Salidas	
LE	BI	LT	D C B A	a b c d e f g	Visualiz.
0	1	1	0 0 0 0	1 1 1 1 1 1 0	0
0	1	1	0 0 0 1	0 1 1 0 0 0 0	1
0	1	1	0 0 1 0	1 1 0 1 1 0 1	2
0	1	1	0 0 1 1	1 1 1 1 0 0 1	3
0	1	1	0 1 0 0	0 1 1 0 0 1 1	4
0	1	1	0 1 0 1	1 0 1 1 0 1 1	5
0	1	1	0 1 1 0	0 0 1 1 1 1 1	6
0	1	1	0 1 1 1	1 1 1 0 0 0 0	7
0	1	1	1 0 0 0	1 1 1 1 1 1 1	8
0	1	1	1 0 0 1	1 1 1 0 0 1 1	9

Por supuesto que de la tabla de verdad, solo tomé lo que me interesa, el resto lo dejé de lado, también se puede notar la configuración de los otros pines del integrado...

El último componente del que hablaremos, es el muy conocido PIC16F84, con el cual nos estamos familiarizando de a poco.

Ahora veamos como es el circuito que vamos a utilizar...



Lo que nos toca ver, es como programar el micro, yo lo haré utilizando la interrupción por el pin RB0, y en él estará conectado el pulsador, y del puerto A usaré los 4 primeros Bits para enviar el dato al decodificador. Ahora, si lo que vamos a hacer es un contador, necesitamos un registro para contar las veces que este se activa, o bien podemos hacer un incremento directamente en el puerto A, yo lo haré de esta última forma.

Un pequeño detalle antes de pasar a la siguiente página...

Si te diste cuenta, estamos utilizando 4 bits para enviar el dato al decodificador, y con 4 bits puedes contar hasta 15 (1111), y este decodificador solo reconoce los datos hasta el 9 (1001), y cuando pase a 1010 (10) el display se apagará, ya que será un dato que no reconoce, cosa que deberemos tener en cuenta al programar.

Una solución sería verificar la cuenta, y cuando llegue a nueve reiniciarla en cero, bueno, pero eso lo veremos en la siguiente página...

Ok, ahora presta atención al código que viene.

:: Microcontroladores PIC - Parte IV - Capitulo 2

Antes quiero aclarar una cosa, para evitarle problemas a aquellos que no se animan a modificar el archivo P16F84.INC, les muestro una opción, ya que en la red encontrarán otros tutoriales o códigos que utilicen este archivo sin modificaciones.

La idea, es crear una copia de este archivo y renombrarlo, por ejemplo P16F84luis.INC (ponle el nombre que mas te guste...) luego le haces las modificaciones a este archivo.

Bien, ya lo advertí, ahora vamos por el código...

```
;---------------Encabezado-------------

      LIST    P=16F84
      #include       <P16F84luis.INC>

;-------Configuración de puertos-------

      ORG     0x00
      GOTO    inicio
      ORG     0x04
      GOTO    ISR
      ORG     0X05

inicio  BSF     STATUS,RP0      ; configurando puertos
      MOVLW   0x10
      MOVWF   TRISA          ; RA0-RA3 = SALIDA
      MOVLW   0xFF           ; PORTB = ENTRADA
      MOVWF   TRISB
      BCF     STATUS,RP0

;-------Habilitación de interrupciones-------

      BSF     INTCON,GIE      ; habilitamos todas las interrupciones
      BSF     INTCON,INTE     ; que sean interrupciones externas

;-------Programa Principal-------

      CLRF    PORTA
espera  SLEEP
      GOTO    espera         ; El micro pasa a bajo consumo

ISR     MOVF    PORTA,W        ; pasamos lo que hay en PORTA a W
      XORLW   B'1001'        ; compara para saber si terminó la cuenta
      BTFSC   STATUS,Z               ; si no terminó salta una linea
      GOTO    reini          ; y si terminó irá a reiniciarla
      INCF    PORTA,F        ; incrementa en 1 PORTA y lo retiene
      BCF     INTCON,INTF     ; borro bandera de interrupción
      RETFIE                 ; regresa al modo SLEEP
reini   CLRF    PORTA
      BCF     INTCON,INTF     ; borro bandera de interrupción
      RETFIE
;-----------------------------------------
      END
;-----------------------------------------
```

Descripción

Y como siempre, sólo aquello que está en rojo, ya que lo demás lo conocemos desde sus inicios.

`#include  <P16F84luis.INC>`

Respecto a esto no diré nada, ya lo mencioné al comenzar esta sección, vamos por lo otro.

Al configurar TRISA con 0x10 hemos dejado RA4 como entrada, de tal modo que enviemos lo que enviemos al registro PORTA, RA4 no será alterado.

De TRISB, bueno, si bien utilizaré sólo uno de sus pines, configuré todo el puerto B como entrada.

Luego viene la habilitación de interrupciones, la general (GIE), y la que corresponde al pin RB0 (INTE)

Lo siguiente, es limpiar el PORTA, para empezar la cuenta en cero, asi que...

```
CLRF    PORTA
```

y el Display muestra cero "0".

Luego ponemos al micro en espera de la interrupción, con la instrucción SLEEP

Ahora biene el gran secreto, La ISR o Rutina de Servicio de Interrupciones...

Les recuerdo que nuestro decodificador cuenta sólo hasta 9, es decir que si envían 10 por el puerto A el Display no mostrará nada, por lo tanto, habrá que reiniciar la cuenta, si el puerto A llega a 9 (B'1001') el próximo pulso deberá enviar cero al display para reiniciar la cuenta.

```
ISR     MOVF    PORTA,W         ; pasamos lo que hay en PORTA a W
        XORLW   B'1001'         ; compara para saber si terminó la cuenta
        BTFSC   STATUS,Z                ; si no terminó salta una linea
        GOTO    reini           ; y si terminó irá a reiniciarla
        INCF    PORTA,F         ; incrementa en 1 PORTA y lo retiene
        BCF     INTCON,INTF     ; borro bandera de interrupción
        RETFIE                  ; regresa al modo SLEEP
```

Cuando se presione el pulsador, se generará una interrupción, eso significa que saldrá del modo **SLEEP** para pasar a este trozo de código.

Teniendo en cuenta lo dicho anteriormente, lo que haremos será pasar lo que hay en PORTA al registro w, y luego compararlo con 1001 (9 en el display). Si aún no llegó a 9 saltamos una línea, incrementamos PORTA (INCF PORTA,F) y lo guardamos en el mismo registro, aquí utilicé F (recuerda que antes lo indicábamos con **0** o con **1**, y como estamos utilizando nuestro P16F84luis.INC, pues la cosa se pone más clara), luego borramos la bandera de interrupción y regresamos al modo sleep.

ok. Supongamos ahora, que la cuenta ya terminó...

En este caso, nuestro Display muestra 9, y PORTA está en **00001001**, si es así, cuando hagamos **xorlw** con **00001001**, por ser el mismo valor, la bandera de cero **Z** del registro STATUS, se pondrá en **1**, pues bien, eso significa que la cuenta terminó, por lo tanto habrá que reiniciarla, asi que hacemos un GOTO a la etiqueta **reini**

```
reini   CLRF    PORTA
        BCF     INTCON,INTF     ; borro bandera de interrupción
        RETFIE
```

Lo único que tenemos que hacer aquí, es poner PORTA a cero, el decodificador lo tomará y pondrá el display en CERO, luego limpiamos la bandera de interrupción y regresamos al modo SLEEP.

Bien, Respecto a lo de limpiar PORTA cuando se inicia el código, lo hice de tal modo que puedas reiniciar la cuenta cuando lo desees, simplemente presionando el pulsador de RESET, personalmente creo que este pulsador debería estar siempre en todos los circuitos, y además es importante tenerlo en cuenta, aunque no lo estuviera.

Bueno..., Este fue el modo sencillo para enviar datos a un Display

Ahora lo vamos a complicar un poquitin más, te animas...???

:: Microcontroladores PIC - Parte IV - Capitulo 3

El Registro PCL

Antes de continuar, veamos como trabaja el micro cuando se encuentra ante una serie de instrucciones.

Please...!!!, abstenerse todos los entendidos en el tema, que esto es para duros como yo...!!! (ya lo advertí...)

Existe un registro, llamado PCL, ubicado en la posición 0x02 en el banco de memoria, tiene mucho que ver con el flujo del programa, puesto que le asigna un número a cada línea de código.

Todo empieza con la primera instrucción, esta tiene una posición indicada con un número en el registro PCL, ok. cuando accede a esa posición, se lee la instrucción, se decodifica, y luego se ejecuta, una vez echo esto, el reloj del micro incrementa al contador de programa (PCL) en un unidad, esto hace que el PCL apunte a la segunda instrucción, ahora se lee esta segunda instrucción, se decodifica y también se ejecuta. Nuevamente, el reloj del sistema incrementa el PCL para que apunte a la tercera instrucción, la decodifique y la ejecute. Este proceso se repite hasta que termina el programa (es decir, cuando encuentra un END).

Se habrá entendido...?

Ahora te lo mostraré con una pequeña animación, aquí el PCL está representado por una flecha (repito, es un número que indica la posición de cada línea de código), observa, (bueno, actualiza la página)...

Bien, de eso se trata, imagínate que te encuentras en un...

GOTO allá

GOTO, es saltar
allá, es la etiqueta de un procedimiento.

Es decir, saltar o ir a la dirección donde se encuentra la etiqueta allá, y continuar desde allí..., es decir que al utilizar esta instrucción estas direccionando la secuencia del programa a otra posición.

Piensa, que si Assembler no nos permitiría utilizar etiquetas, deberíamos decirle la dirección del PCL donde se encuentra ese procedimiento, y vaya Dios a saber que número le corresponde a esa dirección, claro que... en realidad no tienes que preocuparte por ello, para eso están las etiquetas.

Te preguntarás que demonios tiene que ver todo esto con lo que estamos viendo, pues bueno, no desesperes, sólo nos falta una cosa más...

Las Tablas:

Me imagino que sabes lo que es una tabla, bueno, una tabla es algo como esto...

Cont. de Programa	ISNT.	DATO
PCL=11 »	RETLW	11000000
PCL=12 »	RETLW	11100001
PCL=13 »	RETLW	00001111
PCL=14 »	RETLW	00111001

Te preguntarás por el contenido de esta tabla, bueno, hablemos de ello...

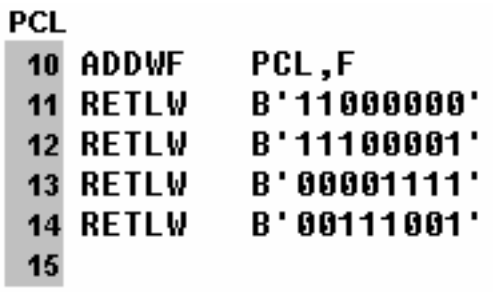
En esta tabla, cada línea horizontal, es una línea de código, y la dirección de cada línea, está dada por el valor

del PCL (el contador de programa), suponte ahora el siguiente código...

RETLW 00001111

RETLW, es retornar cargando **W** con el **L**iteral **00001111**, el problema es que para llegar a esta instrucción deberías pasar por encima de las dos líneas anteriores. La pregunta es, ¿Como se hace eso...?

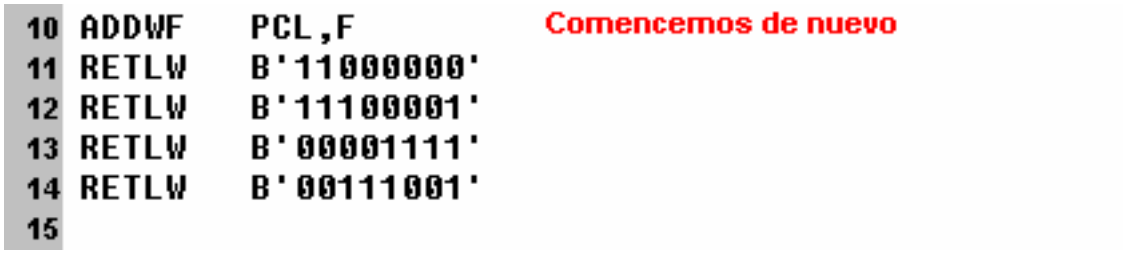
Para entenderlo mejor, grafiqué la misma tabla, pero sin las lineas de separación, también incluí el PCL y le puse un número de orden en decimal (cualquiera...), esto es sólo a modo explicativo ok...?, observa...



La primera instrucción **ADDWF PCL,F** indica que se le debe sumar al registro PCL, lo que hay en W. Con F, le indicamos que guarde el resultado en el mismo registro PCL, es decir...

PCL = PCL + W

El acceso a la tabla lo haremos a travéz de W, le cargamos un valor y llamamos a la tabla, justo donde al PCL se le suma el valor de W, préstale mucha atención a la siguiente animación, creo que es más fácil de entender...



Fíjate que en este ejemplo, los accesos a las lineas 11, 12, 13, 14 y 15, se hacen desde la posición 10, la suma con W indica a que línea debe saltar.

Bien, ahora empiezan las advertencias...

- El registro W es de 8 bits, por lo que el máximo valor será 255, ese será el salto más largo que puedas dar.
- W no debe superar la cantidad de elementos de la tabla, la del ejemplo anterior tiene 4 elementos por lo tanto el valor máximo de W será 3.
- El acceso a la tabla, se hace sólo para tomar el valor que se busca y regresar al programa principal.
- Los comentarios en una tabla, no son tenidos en cuenta por el PCL, estos son ignorados ...

Bien mis queridos amigos, si lograron comprender bien lo de las tablas, los invito a continuar, que ahora viene lo mejor, aplicaremos todo lo visto en esta sección...

:: Microcontroladores PIC - Parte IV - Capitulo 4

Para no aburrirlos con lo del pulsador, haré que el micro envíe unas cuantas señales por su propia cuenta con un pequeño retardo, lo que haremos será una cuenta regresiva de 5 a 0 y luego haremos que escriba LUIS. (con el puntito incluído), que original, no...?

Como esta vez lo haremos sin decodificador, las cosas se verán totalmente distintas, se parecerá más a un secuenciador que a otra cosa...

El efecto que busco conseguir es este...

Bien, comencemos...

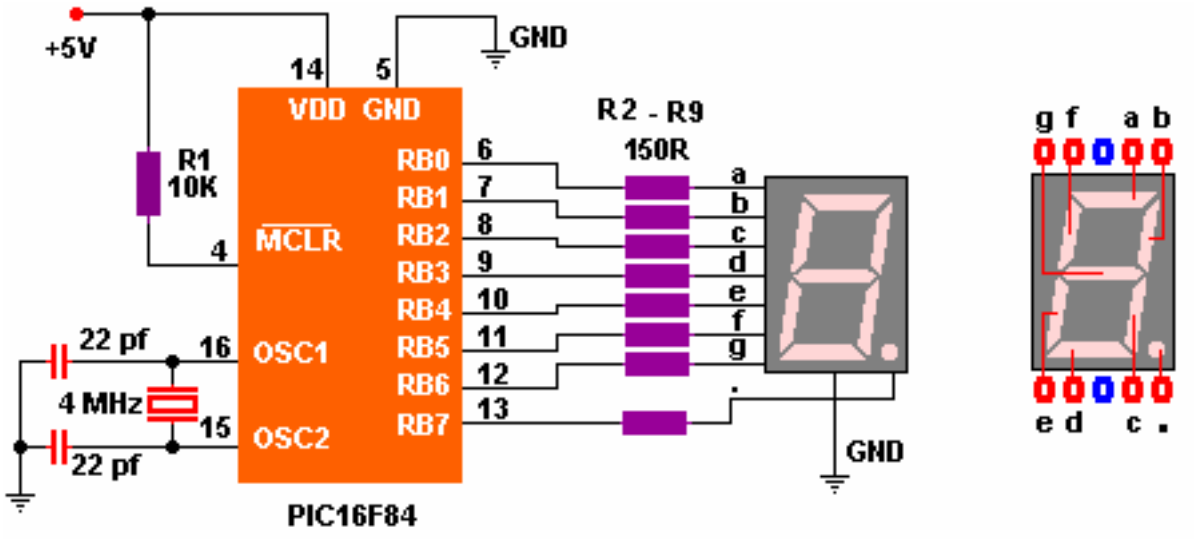
Trabajando directamente con el Display (sin decodificador)

Esta vez, el decodificador, deberemos crearlo nosotros, por medio de código, y el encendido de los segmentos del Display, se hará activándolos desde el micro. Para que tengas una idea, cuando el micro se encienda por primera vez, el display deberá encender los 5 segmentos que corresponden al número **5**, y luego comenzar la secuencia.

Primero veamos lo que necesitamos...

De componentes, sólo el Display de cátodo común, unas cuantas resistencias de 150 ohm y el micro, ya que todo se hará por programa. Ahora pensemos un poco en los pines del micro que utilizaremos...

Como no haremos entradas de señal, dejaremos el puerto A libre. Del puerto B, utilizaremos los 7 pines más bajos (RB0 a RB6) para activar los segmentos del display, y RB7 para el punto. Bien, eso será para la configuración de los pines del micro, ahora veamos el esquema del circuito...



Nuevamente incluí la asignación de las letras a cada segmento, para que no te pierdas.

Se viene lo mejor, "El programa"...

Como haremos una secuencia de caracteres (letras y números) Necesitamos una rutina de retardo que me permita visualizar esa información, también nos hace falta un contador para saber que caracter se mostró en el display y cual es el que sigue, de hecho, a cada caracter le corresponde un código, adivina donde se encuentra ese código...?

siiiiiiii, en una tabla, esta tabla debe contener el código para los números; 5, 4, 3, 2, 1 y 0, mas los caracteres L, U, I, S.

Que tal...?

Ya tenemos todo lo que necesitamos para comenzar, asi que vamos por el código

:: Microcontroladores PIC - Parte IV - Capitulo 5

Código para el Control del Display sin Decodificador

En el encabezado incluimos nuestro ARCHIVO.INC y como variables incorporamos **reg1, reg2** y **reg3** para el retardo, más la variable **cont** que controlará la cuenta para incrementar el PCL por medio de W.

En la configuración de puertos, habilitamos PORTB como salida, y comenzamos con la programación.

```
;---------------Encabezado-------------

      LIST   P=16F84
      #include       <P16F84luis.INC>

;----------- Variables utilizadas ----------

reg1   equ    0x0D          ; 3 registros para el retardo
reg2   equ    0x0E
reg3   equ    0x0F
cont   equ    0x10

;---------Configuración de puertos----------

      ORG    0x00
      GOTO   inicio
      ORG    0x04
      ORG    0X05
inicio  BSF    STATUS,RP0      ; configurando puertos
      CLRF   TRISB          ; PORTB = SALIDA
      BCF    STATUS,RP0

;------------- Programa Principal ----------

reini  CLRF   cont          ; pone el contador a 0
      MOVF   cont,W         ; pasa el contador a w (índice)
      CALL   tabla          ; llama a la tabla
      MOVWF  PORTB          ; pasa el dato obtenido a PORTB
      CALL   retardo

disp_  MOVF   cont,W
      XORLW  B'1001'        ; verifica si el contador llegó a 9
      BTFSC  STATUS,Z               ; si no es así salta una línea
      GOTO   reini          ; si llegó a 9 lo atiende en reini
      INCF   cont,F         ; incrementa el contador
      MOVF   cont,W         ; pasa el contador a w (índice)
      CALL   tabla          ; llama a la tabla
      MOVWF  PORTB          ; pasa el dato obtenido en la tabla a PORTB
      CALL   retardo
      GOTO   disp_

;--------------- Tabla --------------------

tabla  ADDWF  PCL,F         ; se incrementa el contador de programa
      ;display         . gfedcba    segmentos de los leds del display
      RETLW  B'01101101'    ; código para el 5
      RETLW  B'01100110'    ; código para el 4
      RETLW  B'01001111'    ; código para el 3
      RETLW  B'01011011'    ; código para el 2
      RETLW  B'00000110'    ; código para el 1
      RETLW  B'00111111'    ; código para el 0

      RETLW  B'00111000'    ; código para el L
      RETLW  B'00111110'    ; código para el U
      RETLW  B'00000110'    ; código para el I
      RETLW  B'11101101'    ; código para el S.

;-----------Rutina de Retardo-----------

retardo movlw  30           ; Aquí se cargan los registros
      movwf  reg1           ; reg1, reg2 y reg3
tres   movlw  20            ; con los valores 30, 20 y 35
      movwf  reg2
dos    movlw  35
      movwf  reg3

uno    decfsz reg3,1        ; Aquí se comienza a decrementar
      goto   uno
```

```
        decfsz  reg2,1
        goto    dos
        decfsz  reg1,1
        goto    tres
        retlw   00          ; regresare del retardo
;-----------------------------------------
        END
;-----------------------------------------
```

Descripción

Vamos por el programa principal...

```
reini   CLRF    cont        ; pone el contador a 0
        MOVF    cont,W      ; pasa el contador a w (índice)
        CALL    tabla       ; llama a la tabla
        MOVWF   PORTB       ; pasa el dato obtenido a PORTB
        CALL    retardo
```

En la primer linea, ponemos el contador a cero, en la segunda, lo pasamos al registro **W**, es decir W=00000000 y nos vamos con este valor a la tabla, veamos que ocurrirá allí...

```
tabla   ADDWF   PCL,F       ; se incrementa el contador de programa
        ;display       . gfedcba    segmentos de los leds del display
        RETLW   B'01101101'    ; código para el 5
```

ADDWF PCL,F es sumarle al PCL lo que trae W, y como W=00000000, pues PCL seguirá siendo igual a PCL, y pasará a la siguiente instrucción...

RETLW B'01101101', recuerda que la linea de comentario no es tenida en cuenta. En esta línea, se carga w con **01101101**, y como se trata de una instrucción de retorno, regresa al lugar de donde vino, es decir a...

```
MOVWF   PORTB   ; pasa el dato obtenido a PORTB
CALL    retardo
```

Aquí se pasa el valor de **W** a **PORTB** y se visualiza **5** en el Display, luego se hace un retardo, y cuando termina...

```
disp_   MOVF    cont,W
        XORLW   B'1001'     ; verifica si el contador llegó a 9
        BTFSC   STATUS,Z            ; si no es así salta una línea
        GOTO    reini       ; si llegó a 9 lo atiende en reini
        INCF    cont,F      ; incrementa el contador
        MOVF    cont,W      ; pasa el contador a w (índice)
        CALL    tabla       ; llama a la tabla
```

Cargamos W con lo que hay en el contador, y luego, lo que nos toca hacer, es averiguar si ya se mostraron todos los valores que figuran en la tabla, para eso utilizamos la instrucción de comparación **XORLW** con 9 en binario (00001001) puesto que son 10 los elementos de la tabla (del elemento 0 al elemento 9), la instrucción XORLW ya la vimos anteriormente, pero sirve recordarla.

Piensa que si el contador está en 1001 (9), ya mostro todos los elementos de la tabla, y la comparación XORLW dará como resultado 00000000 y la bandera de cero (**Z**) del registro STATUS se pondrá en **1**, de lo contrario permanecerá en **0**, ahora viene la pregunta...

```
BTFSC   STATUS,Z
```

Está en cero la bandera Z del registro STATUS...?, si es así, aún faltan elementos por mostrar, entonces salta una línea, y allí...

```
        INCF    cont,F      ; incrementa el contador
        MOVF    cont,W      ; pasa el contador a w (índice)
        CALL    tabla       ; llama a la tabla
```

Y este trozo de código se repetirá hasta que se muestren todos los elementos.

Bien. Suponte ahora, que la cuenta ya terminó, y se mostraron todos los elementos, eso significa que "cont=1001", cuando llegue a la comparación (XORLW) el resultado SÍ dará **00000000**, la bandera **Z** se pondrá en **1**, y cuando llegues a la pregunta...

```
BTFSC   STATUS,Z
```

Está en cero la bandera Z del registro STATUS...?, la respuesta será NO, por lo tanto se mostraron todos los elementos de la tabla, y no se realizará el salto, es decir que pasará a...

```
GOTO    reini
```

y bueno, allí comenzará todo de nuevo...

Bien mis queridos amigos, espero que les haya servido de ayuda este tutorial, yo lo hice con algo sencillo, para que puedan interpretar la forma de trabajar con estos dispositivos. Imagino que mas de uno, tiene proyectos en los cuales puede incorporarlo, o tiene las intensiones de desarrollar uno nuevo con todos estos chiches, que más dá, ahora queda en sus manos, por lo pronto yo me iré a jugar al Mythology, jejeje

BYE...!!!

:: Microcontroladores PIC - Parte IV - Capitulo 6

Parece que esto del Mythology no es lo mío, siempre pierdo...!!!, en fin, veamos con que vamos a seguir...

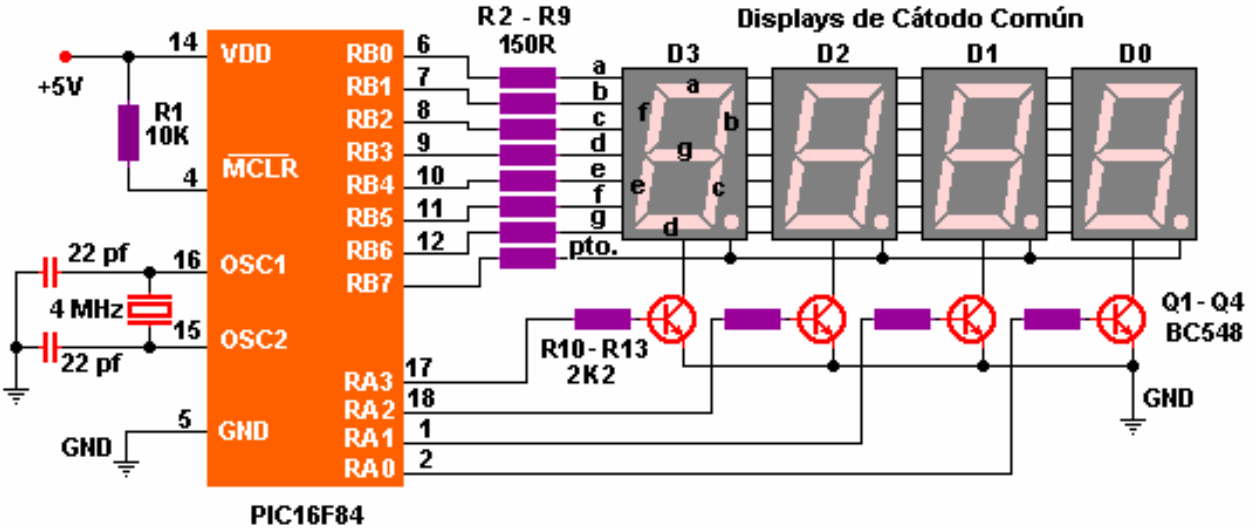
Ah..!!, si..., vieron que interesante fue lo anterior...?, bueno, con las 8 salidas que tiene el micro, nos la arreglamos para manejar un display y activar sus segmentos para mostrar lo que se nos ocurrió, bueno, lo que se me ocurrió.

Imagínate, que pasaría si quisieramos encender o trabajar con 2 displays, la cosa se complica, a demás no tenemos 16 pines en el micro para los dos displays, y si queremos manejar 4...? uff...!!!, peor todavía...!!!

Bueno, también hay una solución, en este caso la idea es multiplexar las señales enviadas por el micro.

Te preguntarás que es eso de multiplexar, Multiplexar es comoooooo, multiplicar, si, es algo así. Algo de esto vimos en el proyecto "secuenciador de 32 canales controlado por PC", claro que allí utilizamos un integrado que se encargaba de mantener la señal enviada por el pc para cada grupo de 8 datos, aquí la cosa será distinta, ya que será el micro quien administre el encendido de cada display y sus segmentos (lo cual se hace por programa).

Para entenderlo mejor, veamos el circuito que vamos a utilizar...



Si prestas atención, el Puerto B se utiliza para enviar los datos a mostrar en cada display, mientras que por el Puerto A seleccionas el display que mostrará ese dato.

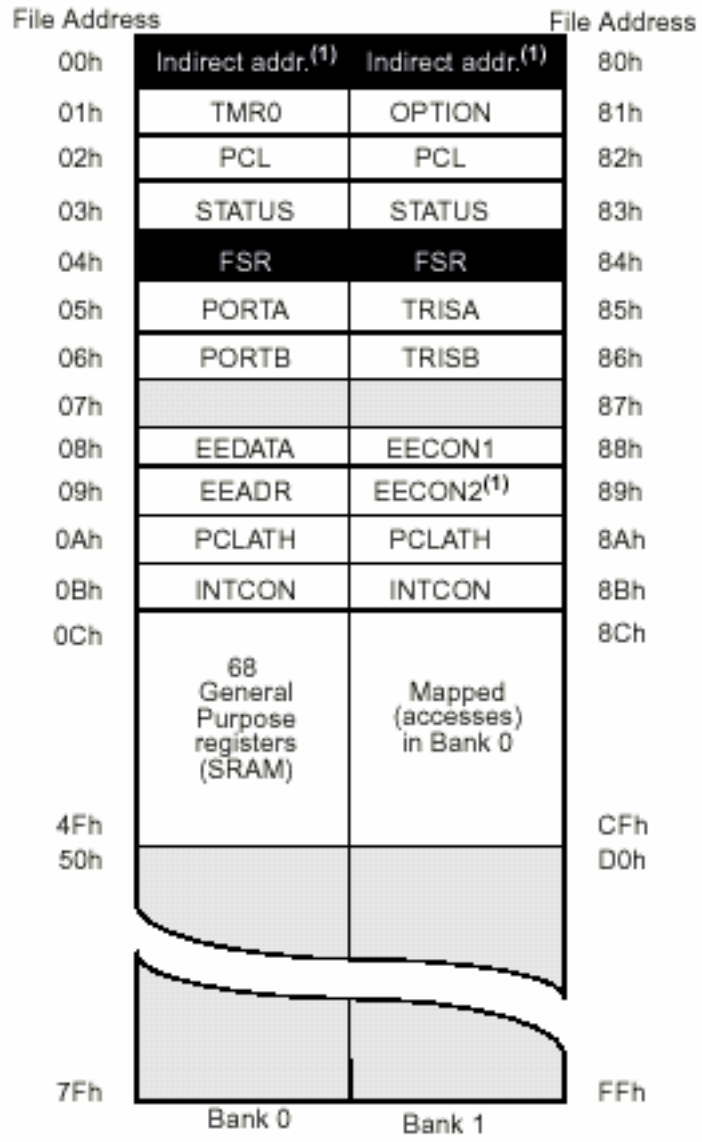
Supongamos que quiero mostrar cero "**0**" en cada Display, pues muy fácil, pongo el puerto B en **00111111** (código para el cero), y activo ahora los transistores conectados en el puerto A, haciendo una secuencia de RA0 a RA3, pero sabes cual es el problema...?, que verás correr el cero de un Display a otro, para solucionar este problema, hagamos lo siguiente, realicemos la secuencia tan rápido, que el observador no note el momento en que cambias de display, por lo tanto vería todos los displays mostrando cero, que picardía no...!!! ;))

Justamente se trata de eso, ahora, si quisiera mostrar LUIS, enviaría "L", "U", "I" y "S" tan rápido como sea posible, de tal modo que nadie note el cambio de display que estoy haciendo para mostrarlo, el micro lo hará tan rápido, que tu verás...

Muy bien, ya está claro lo que haremos, nos falta ver cómo...!!!, para ello vamos a recurrir a un par de registros especiales, de los cuales no hablamos mucho, es más, creo que no hablamos nada de ellos, así que, nos tomamos un tiempo para ver de que se trata...

:: Microcontroladores PIC - Parte IV - Capitulo 7

Antes de mostrarte los registros de los que hablaremos te traje los bancos de memoria del Micro, en donde los resalté para que puedas notarlo...



Estos 2 registros, y en algunos casos, junto al registro STATUS, pueden trabajar en conjunto para hacer un direccionamiento indirecto de la memoria de Datos (memoria RAM). Bien, que es eso del direccionamiento indirecto...?. Para entenderlo mejor estudiemos estos registros...

Registro 04h (FSR)

Es el Registro selector de registros, es un puntero en realidad, Recuerdas aquello de las interrupciones, pues bien, es la misma dirección, la 0x04h, cuando se producía una interrupción, el contador de programa apuntaba a esta dirección, y nosotros le decíamos por donde continuar, o escribíamos ahí lo que debía hacer.

Ok. Ahora utilizaremos el registro contenido en esta dirección para seleccionar otros registros.

Piensa, que si el FSR es un puntero de registros, pues, en un momento, puede apuntar a uno y en otro momento a otro. Ahora, la dirección del registro al que apunta, se copia en un registro llamado **INDF**, y este último registro, se actualiza en cada cambio del registro FSR, ahora... tienes una idea de lo que es el registro INDF...???

Registro 00h (INDF)

Es el registro para direccionamiento indirecto de datos, a pesar de no ser un registro disponible físicamente (esto lo dice la hoja de datos); utiliza el contenido del registro FSR, para seleccionar indirectamente la memoria de datos o RAM. Si la dirección a la que apunta el FSR se copia en INDF, una instrucción aplicada a INDF, determinará lo que se debe hacer con el registro al que apunta.

Veamos un ejemplo, de como trabajan estos dos registros, en colaboración el uno con el otro, y así lo entenderás mejor...

Ejemplo de direccionamiento indirecto

- El Registro 05 contiene el valor 10h
- El Registro 06 contiene el valor 0Ah
- Se Carga el valor 05 en el registro FSR (FSR = 05)
- La lectura del registro INDF retornará el valor 10h
- Se Incrementa el valor del registro FSR en 1 (FSR = 06)
- La lectura del registro INDF retornará el valor 0Ah.

Está mas claro verdad...???

Veamos otro ejemplo pero en código. Lo que hace este miniprograma, es borrar el contenido de la memoria RAM entre 0x20-0x2F utilizando direccionamiento indirecto.

```
      ...
      MOVLW   0x20    ; inicializa el puntero
      MOVWF   FSR     ; a la RAM
siguiente     CLRF    INDF    ; borra el registro INDF
      INCF    FSR     ; incrementa el puntero
      BTFSS   FSR,4   ; terminó ?
      GOTO    siguiente       ; NO, borra el siguiente
SIGUE   ...           ; SI, continúa con el programa
```

Veamos, Primero cargamos W (W=0x20), luego se lo pasamos al FSR, ahora el FSR apunta al registro 0x20, INDF también.

Borramos el registro INDF (lo ponemos a 00000000), en realidad es el registro 0x20 el que estamos poniendo a 00000000, luego incrementamos en uno el registro FSR, es decir, apunta a 0x21, adivina a quién apunta INDF...?, exactamente..., a 0x21.

Ahora viene la pregunta... El Bit4 de FSR está en uno...?, si es que NO, regresa a **siguiente** y borra INDF (está borrando el contenido de 0x21), ahora incrementa FSR (FSR=0x22=INDF), y vuelve a preguntar, como la respuesta es NO, borra INDF (0x22) y nuevamente incrementa FSR, y bueno, así, hasta que FSR llega a 0x2F, en donde la respuesta a la pregunta es SÍ, y salta una línea para continuar con el flujo del programa.

Viste que bueno que está..., imagínate todas las aplicaciones en que los puedes utilizar, ok. les comento que estos ejemplos fueron extraídos de la hoja de datos del PIC16F84, y creo que están bastante entendibles.

De acuerdo, todo lo que vimos hasta el momento, será lo que aplicaremos para hacer un programa que controle 4 displays.

Listos...???

Vamos por lo que sigue...

:: Microcontroladores PIC - Parte IV - Capitulo 8

Volvamos a lo nuestro, y analicemos el programa por partes o en módulos, luego veremos si es necesario un diagrama de flujo...

Primero el encabezado con nuestro archivo .inc para hablar en términos de C, Z, W, F, etc. y la definición de variables...

```
;---------------Encabezado-------------------

      LIST    P=16F84
      #include       <P16F84luis.INC>

;-------- Variables a utilizar --------

ret1    equ     0x0d
ret2    equ     0x0e          ; registros para retardos
rota    equ     0x0f          ; reg. para rotación (cambio de display)
disp1   equ     0x10          ; primer dato a mostrar
disp2   equ     0x11          ; segundo dato a mostrar
disp3   equ     0x12          ; tercer dato a mostrar
disp4   equ     0x13          ; cuarto dato a mostrar
```

Recuerda que lo que haremos sera una secuencia de displays, por lo que es necesario una rutina de retardo, y será muy pequeña, algo como esto...

```
;------- RETARDO -------

retardo MOVLW   0x03
      MOVWF   ret1
dos     MOVLW   0x6E
      MOVWF   ret2
uno     NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      DECFSZ  ret2,F
      GOTO    uno
      DECFSZ  ret1,F
      GOTO    dos
      RETLW   0x00
```

No me voy a gastar explicando el retardo (tema visto anteriormente), sólo lo puse para tenerlo en cuenta, lo que sí rescato de aquí, es el uso de la instrucción **NOP**, que significa no hacer nada (aunque lo que estamos logrando es hacer tiempo). Una cosa más, los registros **reg1** y **reg2** son variables definidas anteriormente.

La configuración de puertos también será sencilla ya que ambos puertos serán de salida uno maneja los datos, y el otro selecciona cada display, entonces...

```
;-------Configuración de puertos-------

reset   ORG     0x00
      GOTO    inicio
      ORG     0x05

inicio  BSF     STATUS,RP0      ; configurando puertos
      CLRF    TRISA         ; portA es salida
      CLRF    TRISB         ; portB es salida
      BCF     STATUS,RP0
```

Habrás notado que en la definición de variables se incluyeron 4 registros llamados **disp1, disp2, disp3** y **disp4**. Estos registros los vamos a utilizar para guardar el valor que se sumará al PCL en la tabla, de tal modo que tome el dato que queremos enviar al display, y como son 4 displays, pues utilizamos 4 registros y le cargamos con la dirección de esos 4 datos, así...

```
; ------- cargando direcc. de datos de la tabla -------
```

```
      MOVLW   0x01
      MOVWF   disp1
      MOVLW   0x02
      MOVWF   disp2
      MOVLW   0x03
      MOVWF   disp3
      MOVLW   0x04
      MOVWF   disp4
```

Y ahora la tabla, será muy pequeña, ya que sólo quiero mostrar mi nombre ;o))

```
;------- TABLA -------

tabla   ADDWF   PCL,F         ; se incrementa el contador de programa
      ;display         . gfedcba    segmentos de los leds del display
      NOP
      RETLW   B'00111000'    ; código para la L
      RETLW   B'00111110'    ; código para la U
      RETLW   B'00000110'    ; código para la I
      RETLW   B'01101101'    ; código para la S
```

Aquí también incluí un NOP, para pasar por encima, cuando el programa venga a buscar el primer dato, y así no empezamos desde cero.

Ahora viene lo más importante, el código principal del programa. Primero borramos el Puerto_A para desactivar todos los transistores (apagar los displays) y luego continuamos con el código.

Hay por allí, un registro llamado "**rota**", que lo vamos a utilizar en el siguiente código para activar los transistores que están conectados a PORTA, de tal modo de seleccionar el display que vamos a encender, puesto que son 4, lo vamos a cargar con "00001000" ó 0x08 para seleccionar uno de los displays, y luego lo haremos rotar, para seleccionar los tres restantes. En la siguiente línea, hacemos que el **FSR** apunte al primer registro **disp1**, y nos preparamos para enviar datos al Display, todo esto en las primeras 4 líneas...

```
; ----------- apaga transistores ----------

      CLRF    PORTA

; ----------- PROG. PPAL ----------

ini     MOVLW   0x08
      MOVWF   rota          ; rota= '00001000'

      MOVLW   disp1
      MOVWF   FSR           ; CARGA FSR CON LA DIRECC. DE disp1

display MOVLW   0x00
      MOVWF   PORTB         ; PORTB=00000000

      MOVF    rota,W
      MOVWF   PORTA         ; PORTA= 00001000

      MOVF    INDF,W        ; lee dato al que apunta FSR (o sea disp1)
      CALL    tabla         ; llama a la tabla
      MOVWF   PORTB         ; pasa el dato al puerto B

      CALL    retardo       ; llama miniretardo
      BTFSC   rota,0        ; rota = 00000000 ???
      GOTO    ini           ; si es así, se vio todo, reinicia

      BCF     STATUS,C            ; carry = 0 (para no afectar rotaciones)
      RRF     rota,F        ; rota display
      INCF    FSR,F         ; apunta al siguiente disp_X
      GOTO    display
```

En las dos primeras líneas de la etiqueta **display** enviamos 00000000 a PORTB (puesto que los display's son de cátodo común, los 4 estarán apagados), y luego seleccionamos el transistor del display de la izquierda, esto lo hacemos poniendo 00001000 en PORTA.

Recuerda que el FSR apuntaba a disp1, y como ya sabemos, INDF también, y cuando leamos INDF, estaremos leyendo disp1, luego lo pasamos a **W**, para seguidamente llamar a la tabla, tomar el dato y mostrarlo en el display seleccionado. Como disp1=1 estaremos tomando el código para la letra **L** de la tabla, y lo estaremos enviando al display de la izquierda.

Bien, ahora hacemos un miniretardo, y al regresar, preguntamos si se terminó de rotar, como recién comenzamos..., aún falta..., Ahora bien, por una cuestión de precaución borramos el Carry del registro STATUS, así no se afecta la rotación, de lo contario cuando terminemos de rotar apareceran cosas raras como

un uno demás, asi que lo borramos y hacemos la rotación a la derecha del registro **rota**, luego incrementamos el FSR (para que apunte al registro **disp2**) y regresamos a **display**

veamos como estan las cosas, **rota=00000100, FSR=disp2=INDF**, ok, eso significa que ahora, con rota seleccionamos el siguiente display, cuando tomemos el dato de INDF, estaremos tomando el dato de disp2, y de la tabla tomaremos el código para la letra **U**, haremos un retardo, verificamos la rotación y si no terminó, seguiremos rotando, incrementamos el FSR para ir por el siguiente dato, y repetimos el ciclo.

Esta vez **rota=00000010, FSR=disp3=INDF**, es decir que esta vez mostraremos la **I**, y seguiremos así hasta mostrar la **S**, cuando esto ocurra, y lleguemos a la pregunta de si terminó la rotación, nos daremos con que SÍ, y entonces saltaremos a **ini**, para repetir la secuencia de displays.

Wowwww...!!!, terminamooooos...!!!, parecía que sería más extenso, pero no, claro que este programita, es con la intensión de mostrar usos y aplicaciones del micro, cada uno sabrá la utilidad que le dará, y para que lo pongan a prueba, les dejo el programa completo...

:: Microcontroladores PIC - Parte IV - Capitulo 9

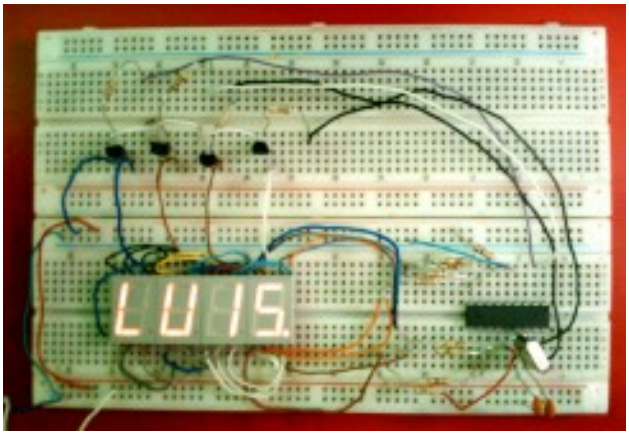
Recuerda, si quieres ensamblar este programa, deberás modificar el encabezado, cambiando el nombre del archivo **P16F84luis.INC** por el que tu tienes, por lo demás no creo que tengas problemas.

Suerte...!!!

```
;---------------Encabezado-------------------

      LIST    P=16F84
      #include       <P16F84luis.INC>

;-------- Variables a utilizar --------

ret1    equ     0x0d          ; utilizado en retardos (milisegundos)
ret2    equ     0x0e          ; utilizado en retardos
rota    equ     0x0f          ; rota el uno para habilitar displays
disp1   equ     0x10          ; primer dígito a mostrar
disp2   equ     0x11          ; segundo dígito a mostrar
disp3   equ     0x12          ; tercer dígito a mostrar
disp4   equ     0x13          ; cuarto dígito a mostrar

;-------Configuración de puertos-------

reset   ORG     0x00
      GOTO    inicio
      ORG     0x05

inicio  BSF     STATUS,RP0      ; configurando puertos
      CLRF    TRISA           ; portA es salida
      CLRF    TRISB           ; portB es salida
      BCF     STATUS,RP0

; ------- carga de registros a mostrar -------

      MOVLW   0x01
      MOVWF   disp1
      MOVLW   0x02
      MOVWF   disp2
      MOVLW   0x03
      MOVWF   disp3
      MOVLW   0x04
      MOVWF   disp4

; ----------- apaga transistores ----------

      CLRF    PORTA

; ----------- PROG. PPAL ----------

ini     MOVLW   0x08
      MOVWF   rota          ; rota= '00001000'

      MOVLW   disp1
      MOVWF   FSR           ; CARGA FSR CON LA DIRECC. DE disp1

display MOVLW   0x00
      MOVWF   PORTB         ; PORTB=00000000

      MOVF    rota,W
      MOVWF   PORTA         ; PORTA= 00001000

      MOVF    INDF,W        ; lee dato al que apunta FSR (o sea disp1)
      CALL    tabla         ; llama a la tabla
      MOVWF   PORTB         ; pasa el dato al puerto B

      CALL    retardo       ; llama miniretardo
      BTFSC   rota,0        ; rota = 00000000 ???
      GOTO    ini           ; si es así, se vio todo, comienza otra vez

      BCF     STATUS,C            ; carry = 0 (para no afectar rotaciones)
      RRF     rota,F        ; rota display
      INCF    FSR,F         ; apunta al siguiente disp_X
      GOTO    display

;------- RETARDO -------

retardo MOVLW   0x03
      MOVWF   ret1
dos     MOVLW   0x6E
```

```
      MOVWF   ret2
uno     NOP
      NOP
      NOP
      NOP
      NOP
      NOP
      DECFSZ  ret2,F
      GOTO    uno
      DECFSZ  ret1,F
      GOTO    dos
      RETLW   0x00

;------- TABLA -------

tabla   ADDWF   PCL,F         ; se incrementa el contador de programa
      ;display         . gfedcba    segmentos de los leds del display
      NOP
      RETLW   B'00111000'     ; código para la L
      RETLW   B'00111110'     ; código para la U
      RETLW   B'00000110'     ; código para la I
      RETLW   B'11101101'     ; código para la S.
;----------------------------------------
      END
;----------------------------------------
```

Sería bueno verlo funcionar, asi que aquí lo tienen...



Es eso simplemente, mostrar un mensaje, y la secuencia entre cada carecter es muy difícil de notar, ya que la velocidad es muy elevada.

Podríamos mejorarlo y hacer que se desplacen los caracteres de un lado a otro, no crees...???, eso lo dejo en tus manos, ya que con todo lo que tienes, puedes hacer lo que se te ocurra, es más, podrías harmarte tus display's con LED's comunes, agruparlos en forma de segmentos y trabajar con ellos, que más, bueno, no se, ya verás que es lo que haces, o te quedarás simplemente con esto...???

Creo que fue suficiente por esta vez, espero que les haya sido de utilidad y que disfruten de todo lo que vimos hasta ahora, no es gran cosa, pero de algo sirve, no les parece...???

Saludos para todos...!!!

R-Luis