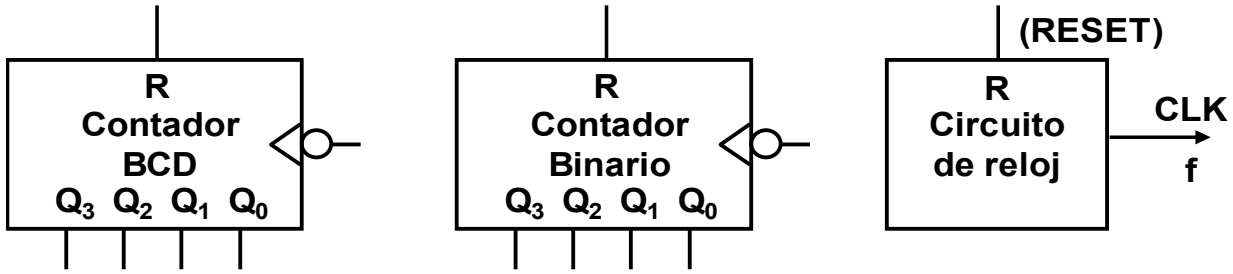
EJERCICIO 1

Si se dispone de un reloj de frecuencia f_{CLK} kHz, se pide:

a) Realizar un circuito que permita obtener una frecuencia de 1Hz

b) Utilizando el reloj de 1Hz, obtenido en el apartado (a) diseñar un temporizador que realice una temporización de T segundos, si se dispone, además del reloj, de los elementos usuales, a saber: fuente de alimentación de 5V, cualquier tipo de puertas lógicas de cualquier número de entradas y bloques MSI combinacionales y secuenciales habituales (entre ellos biestables de cualquier tipo, y contadores binarios de 4 bits, contadores BCD, etc.)
El temporizador se activará al actuar sobre un pulsador. Proponer como conectarlo.
Datos: El único reloj disponible es el proporcionado.

Algunos de los elementos disponibles son:

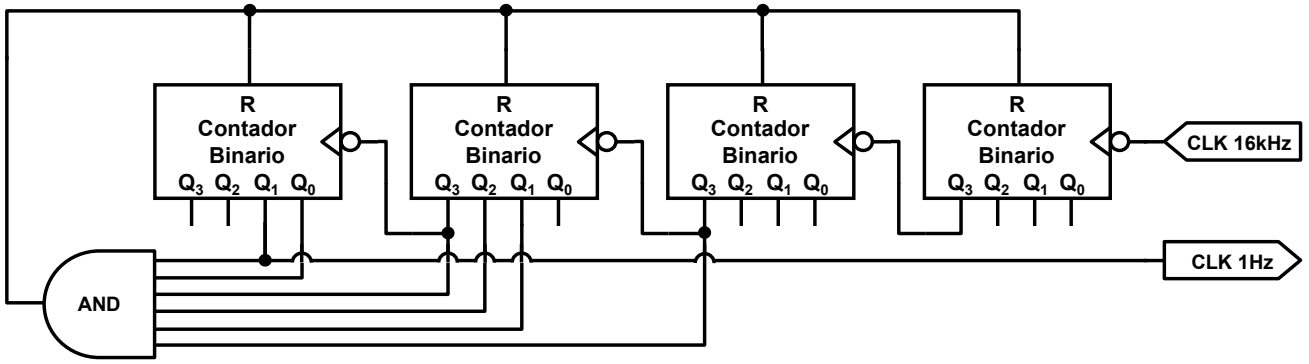


a) Se puede realizar de diferentes formas.

- a.1. Solución óptima: con contadores binarios (requiere el mínimo de contadores):
- a.2. Otra solución válida es utilizar contadores BCD (más fácil de razonar, al utilizar una cuenta "natural" en BCD, aunque utiliza un contador más)
- a.3. Finalmente, se podría realizar mediante divisiones de frecuencia sucesivas. En este caso hay muchas más opciones. Por ejemplo, para f=16000, se puede hacer mediante 16000=16x10x10x10x10, o 16000=50x32x10, o muchas otras formas (no se indica)

a.1. Solución con contadores binarios.
Según los modelos, se tenía:
f_{CLK}=16kHz, 32 kHz, 46 kHz, 64 kHz

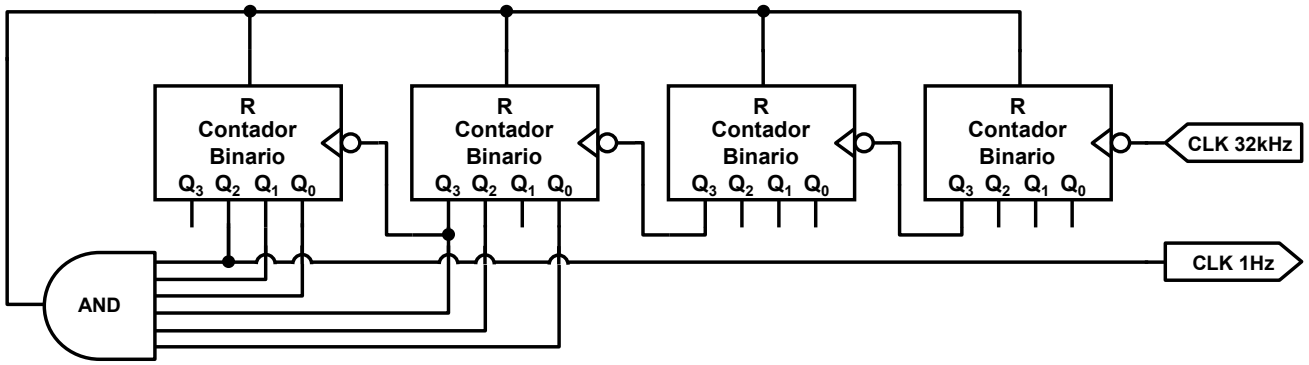
16kHz=16000=8192+4096+2048+1024+512+128=$2^{13}+2^{12}+2^{11}+2^{10}+2^{9}+2^{7}$=
=b'0011 1110 1000 0000'



RESET con 16000 = b'0011 1110 1000 0000'

0 0 1 1 1 1 1 0 1 0 0 0 0 0 0 0

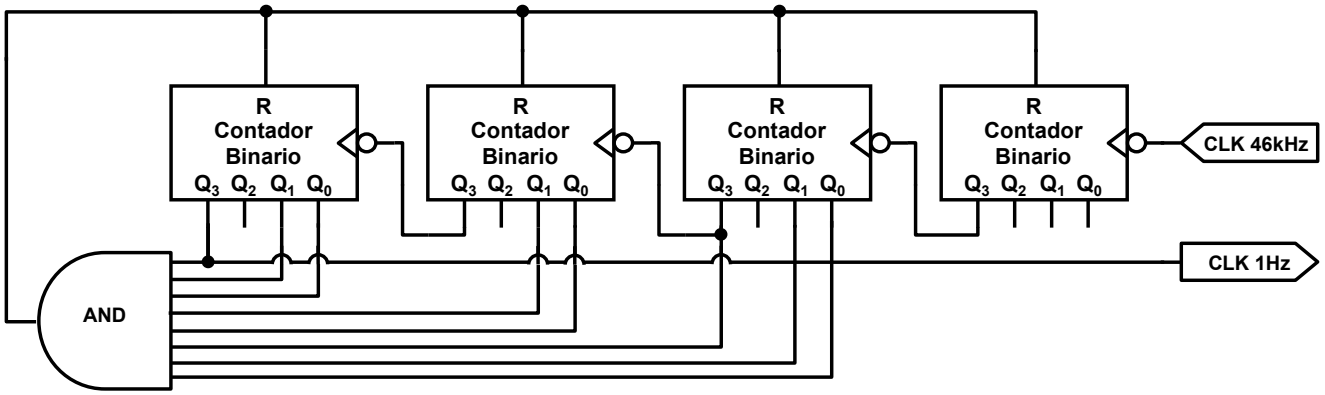
32kHz=32000=16384+8192+4096+2048+1024+256=$2^{14}+2^{13}+2^{12}+2^{11}+2^{10}+2^{8}$=
=b'0111 1101 0000 0000'



RESET con 32000 = b'0111 1101 0000 0000'

0 1 1 1 1 1 0 1 0 0 0 0 0 0 0 0

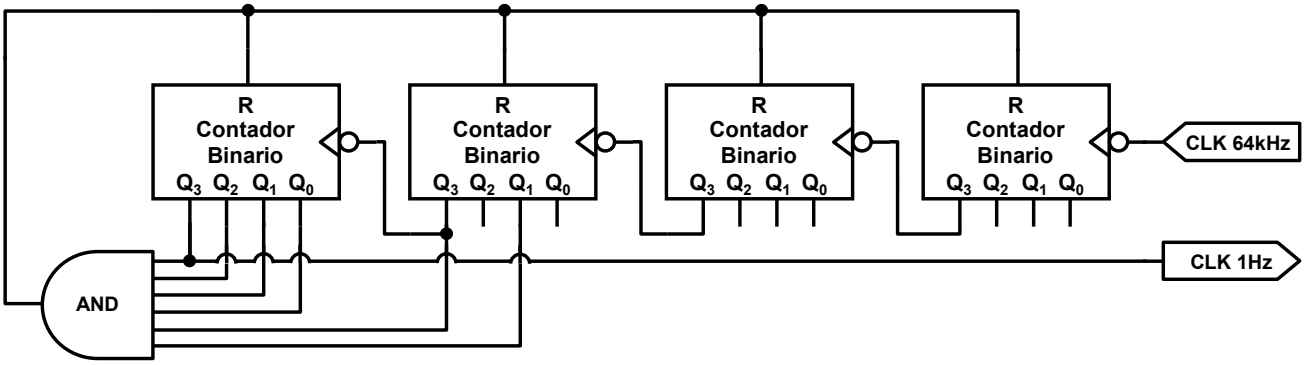
46kHz=46000=32768+8192+4096+512+256+128+32+16=$2^{15}+2^{13}+2^{12}+2^{9}+2^{8}+2^{7}+2^{5}+2^{4}$=b'1011
0011 1011 0000'



RESET con 46000 = b'1011 0011 1011 0000'

1 0 1 1 0 0 1 1 1 0 1 1 0 0 0 0

64kHz=64000=32768+16384+8192+4096+2048+512=$2^{15}+2^{14}+2^{13}+2^{12}+2^{11}+2^{9}$=b'1111 1010 0000 0000'

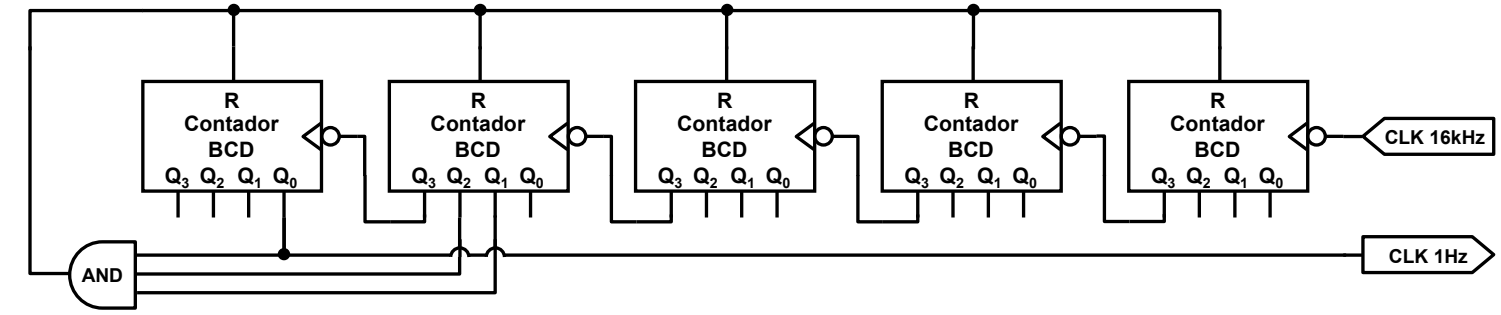


RESET con 64000 = b'1111 1010 0000 0000'

1 1 1 1 1 0 1 0 0 0 0 0 0 0 0 0

a.2. Solución con contadores BCD

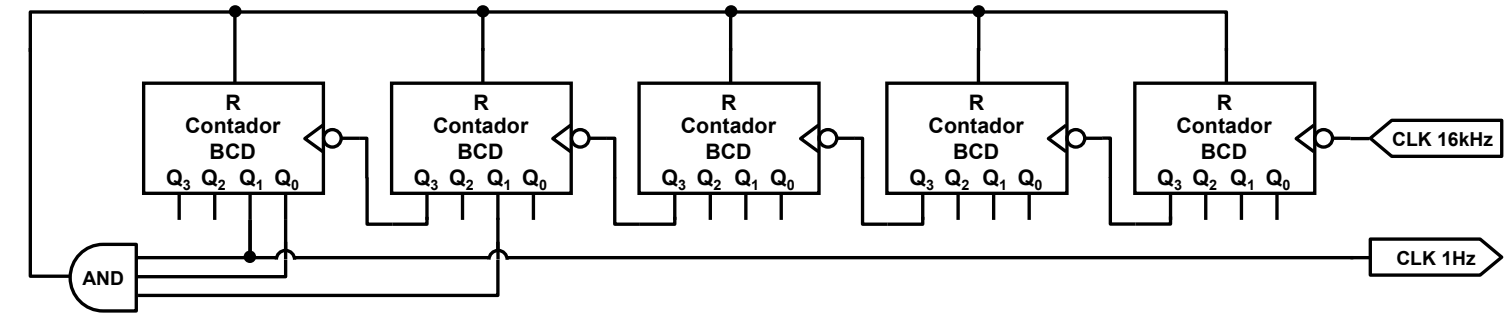
f_{CLK}=16kHz



RESET con 16000

0 0 0 1	0 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0
1	**6**	**0**	**0**	**0**

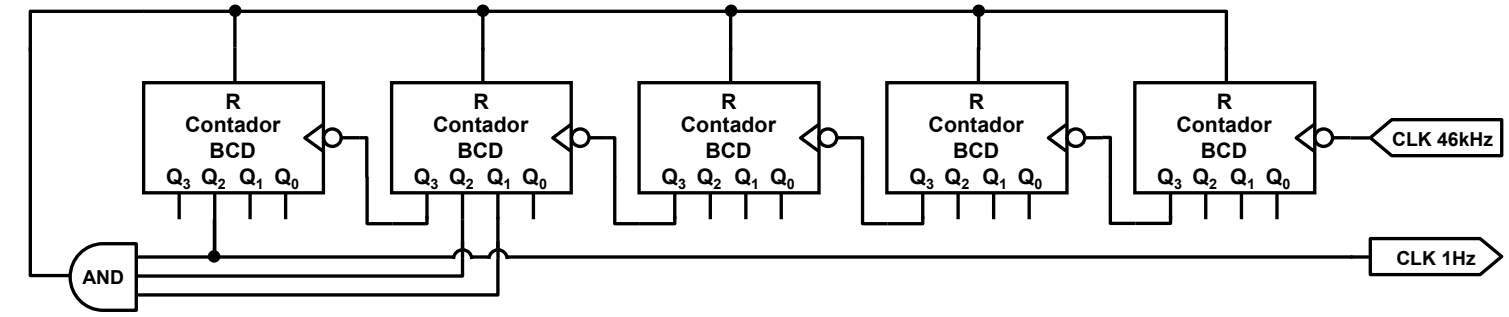
f_{CLK}=32kHz



RESET con 32000

0 0 1 1	0 0 1 0	0 0 0 0	0 0 0 0	0 0 0 0
3	**2**	**0**	**0**	**0**

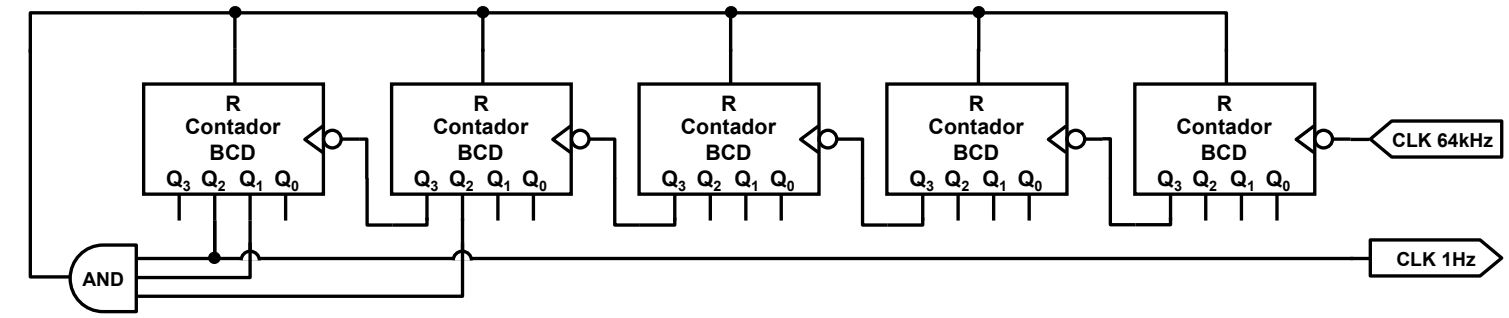
f_{CLK}=46kHz



RESET con 46000

0 1 0 0	0 1 1 0	0 0 0 0	0 0 0 0	0 0 0 0
4	**6**	**0**	**0**	**0**

f_{CLK}=64kHz

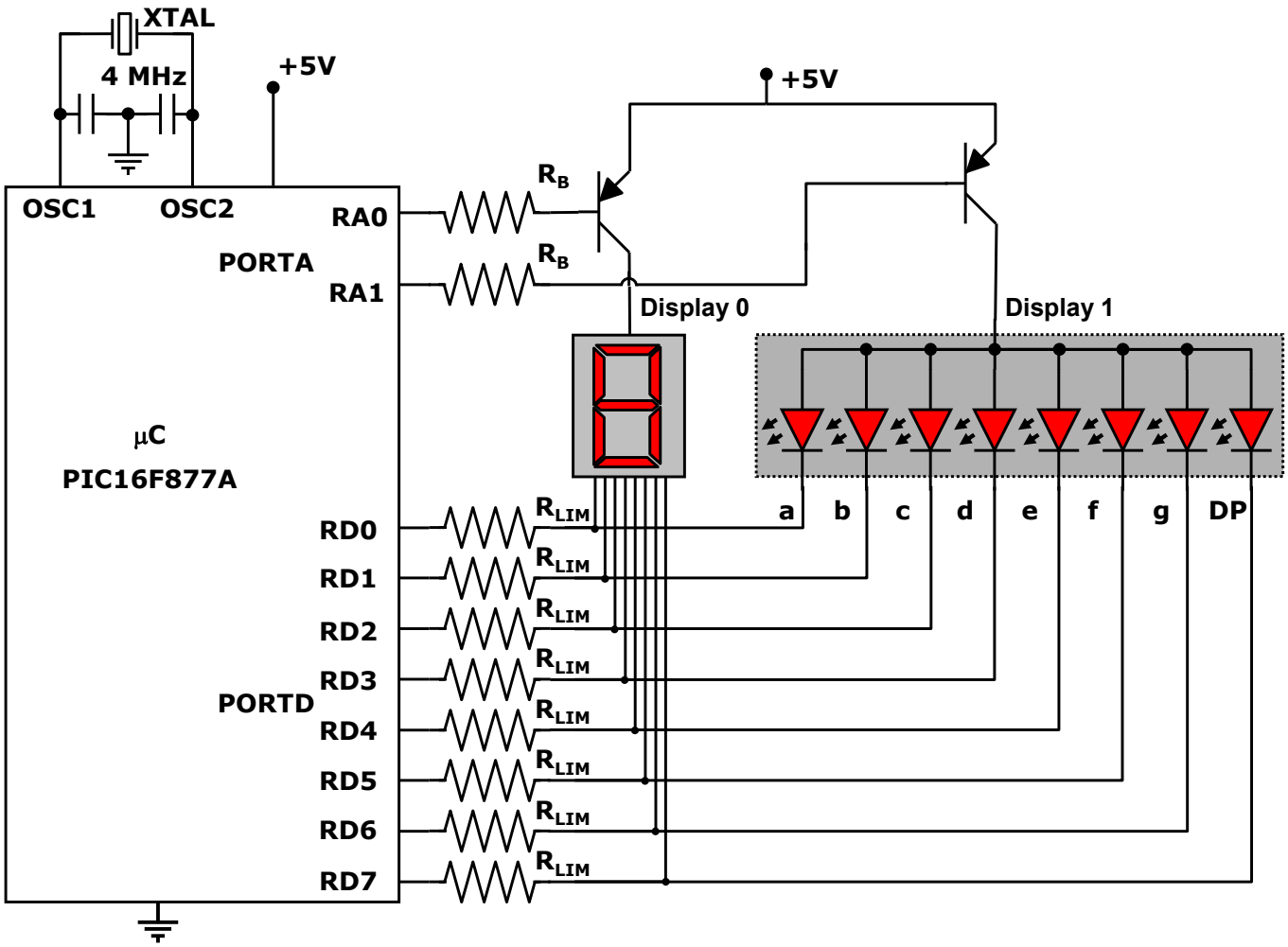


RESET con 64000

0 1 1 0	0 1 0 0	0 0 0 0	0 0 0 0	0 0 0 0
6	**4**	**0**	**0**	**0**

EJERCICIO 2

Se quiere realizar un circuito para realizar el encendido de dos displays desde un microcontrolador PIC16F877A, utilizando el procedimiento de barrido secuencial, para lo que se ha dispuesto el circuito de la figura.



El código a representar en cada display está guardado en las posiciones 0x20 (display0) y 0x21 (display1) (hay otra parte del código, que no se muestra aquí, que se encarga de esta tarea)

La información del valor a mostrar en cada display está codificada directamente en 7 segmentos por lo que el programa a realizar solo debe realizar el encendido de los displays mediante las líneas del PORTA seleccionadas y el volcado de la información de cada display sobre el puerto D. Para cada display, el proceso será:

a) Apagar el display
b) Volcar el contenido de la información a mostrar sobre el puerto D
c) Encender el display
d) Esperar un tiempo RETARDO con el display encendido
e) Apagar el display
f) Pasar al siguiente display, empezando el proceso de nuevo en a)

Para conseguir el funcionamiento deseado, nos han proporcionado el siguiente programa:

```
          LIST p=16F877A          ; Procesador
          INCLUDE <p16f877A.inc>; Incluye definiciones para procesador
                                  ; Ajusta la palabra de configuración
          __CONFIG    _XT_OSC & _WDT_OFF & _PWRTE_ON & _BODEN_ON & _LVP_OFF

DISPLAY0    EQU 0x20              ; Posición para código para el Display0 en 7 segmentos
DISPLAY1    EQU 0x21              ; Posición para código para el Display1 en 7 segmentos

V_RESET     ORG 0x00              ; Vector de RESET
            goto 0x05             ; Salta el vector de interrupción en 0x04
                                  ; Aunque no utilicemos interrupciones, desactivadas tras el RESET
            ORG 0x05
INICIO      movlw b'00000000'     ; Valores iniciales para PORTA y PORTB
            movwf PORTA           ; Desactiva todos los displays
            movwf PORTD           ; Código inicial en PORTD todos los LED apagados
            bsf STATUS,RP0        ; Banco 1
            movlw b'00000000'     ; Puerto A todas las líneas digitales
            movwf ADCON1
            movlw b'11001111'     ; Lineas RA0 y RA1 salidas.
            movwf TRISA
            movlw b'11111111'     ; Puerto D todo salidas
            clrf TRISD
            movlw b'11001110'     ; Inicializa Timer0
            movwf OPTION_REG
            bsf STATUS,RP1        ; Banco 0
                                  ; Inicializa valores a sacar por los displays
                                  ; DISPLAY0 y DISPLAY1 contienen los valores a volcar sobre PORTD
BARRIDO     movf  DISPLAY0, W     ; Carga el valor a representar en 7 segmentos del Display 0
            movwf PORTD           ; Saca el valor del primer digito por PORTD
            bcf PORTA,0           ; Luego activamos el display 0 (display de decenas)
                                  ; activando la salida correspondiente RA0
            call RETARDO          ; Retardo mientras está activo el display 0
            bsf PORTA,0           ; Desactiva el display 0
            movf  DISPLAY1, W     ; Carga el valor a representar en 7 segmentos del Display 1
            movwf PORTD           ; Saca el valor del primer digito por PORTD
            bcf PORTA,1           ; Luego activamos el display 0 (display de decenas)
                                  ; activando la salida correspondiente RA1
            call RETARDO          ; Retardo mientras está activo el display 0
            bsf PORTA,1           ; Desactiva el display 0
            goto BARRIDO          ; Vuelve a empezar otro barrido

; RETARDO. Genera un retardo durante el cual el display está encendido
RETARDO     bcf INTCON,T0IF     ; Ponemos a cero el flag de overflow del Timer0
            movlw d'178'        ; Valor de precarga para conseguir el tiempo previsto
            movwf TMR0          ; Precarga TMR0 iniciando la temporización
ESPERA      btfss INTCON,T0IF   ; Si rebosa el temporizador, salimos
            goto ESPERA         ; Si no rebosa sigue esperando
            return
            END
```

DATOS:

a) El cristal del oscilador para el ciclo de instrucción interno tiene una frecuencia $f_{CLK\text{-}INT}$=4 MHz

b) El tiempo de encendido de cada display se obtiene mediante la subrutina RETARDO y se desea que sea de 15 ms.

Se pide:

a) Calcular, con la actual configuración del temporizador, y la actual subrutina de temporización, cuál sería el tiempo que se temporiza realmente en la subrutina "RETARDO", justificando la respuesta

b) Si el tiempo no es correcto, proponer los cambios a realizar en el programa (tanto en la configuración como en el uso del TIMER TMR0) para conseguir temporizar el tiempo solicitado, justificando la respuesta.

c) Con la conexión del PIC que se indica, analizar si la configuración y el uso posterior de los puertos utilizados es correcta para conseguir el propósito del programa. Justificar si el programa es correcto y, de no ser así, indicar los errores cometidos y los cambios a realizar en el programa para conseguir un correcto funcionamiento, coherente con el esquema eléctrico del circuito, justificando la respuesta

SOLUCIÓN EJERCICIO 2: MODELO CON DOS DISPLAYs ÁNODO COMÚN

PORTD usado para selección de display mediante PNPs a los ánodos comunes (0 enciende, 1 apaga), PORTA saca el código en siete segmentos (0 enciende, 1 apaga)

a) Configuración actual:

Inicialización: OPTION_REG = b '11001110'. Los bits 7 y 6 no nos importan

Bit 5: T0CS=0. Usa el reloj interno de instrucción

Bit 4: T0SE=0. Transición usada si CLK externo: no afecta su valor en este caso

Bit 3: PSA=1. Asigna el prescaler al Watchdog. Por tanto el Timer0 no usa prescaler, y la cuenta es directa (es como si fuera prescaler=1)

Bits <2:0> <PS2:PS0> valen 110. El prescaler del Watchdog es 64, pero no nos afecta su valor para la temporización del Timer0.

Uso del Timer0: Se carga el valor 178 (=precarga)

El tiempo temporizado es entonces:

$Tiempo=[(256-precarga)\cdot Prescaler+2]\cdot T_{INSTRUCCIÓN}$

$T_{INSTRUCCIÓN}=4\cdot T_{CLK}=4/f_{CLK}=4/4Mhz=1\ \mu s$

$Tiempo=[(256-178)\cdot 1+2]\cdot(1\ \mu s)=80\ \mu s$

b) Queremos temporizar 15 ms

En primer lugar, calculamos el máximo valor que se podría temporizar con el reloj interno (único disponible). Para eso ponemos el prescaler al máximo (Prescaler=256) y el valor cargado en TMR0 al menor valor posible, para maximizar la expresión:

$Tiempo=[(256-precarga)\cdot Prescaler+2]\cdot T_{INSTRUCCIÓN}$

$T_{INSTRUCCIÓN}=4\cdot T_{CLK}=4/f_{CLK}=4/4Mhz=1\ \mu s$

El máximo tiempo que se puede temporizar en una sola temporización es:

$Tiempo=[(256-0)\cdot 256+2]\cdot(1\ \mu s)=65538\ \mu s=64{,}538$ ms

Podemos, por tanto, temporizar sin problema los 15 ms. Para conseguirlo:

$Tiempo=[(256-precarga)\cdot Prescaler+2]\cdot T_{INSTRUCCIÓN} \Rightarrow$

$$[(256 - precarga) \cdot Prescaler + 2 = \frac{Tiempo}{T_{INSTRUCCIÓN}} \Longrightarrow$$

$$[(256 - precarga) \cdot Prescaler = \frac{Tiempo}{T_{INSTRUCCIÓN}} - 2 \Longrightarrow [(256 - precarga) = \frac{\frac{Tiempo}{T_{INSTRUCCIÓN}} - 2}{Prescaler} \Longrightarrow$$

$$precarga = 256 - \frac{\frac{Tiempo}{T_{INSTRUCCIÓN}} - 2}{Prescaler} = 256 - \frac{\frac{15ms}{1\mu s} - 2}{Prescaler} = 256 - \frac{15000 - 2}{Prescaler} = 256 - \frac{14998}{Prescaler}$$

Valores posibles:

Prescaler	Precarga	Conclusión
256	197,41≅197	Válido
128	138,82≅139	Válido
64	20,65≅21	Válido
32	-213,68 <0	Este valor de prescaler ya no sirve

Según el valor escogido, se ajusta OPTION_REG y el valor a cargar en el TMR0 en la rutina RETARDO

Por ejemplo, si tomamos Prescaler=64, entonces en TMR0 precargamos 21 en la rutina RETARDO. Comprobamos que la temporización sería:

Tiempo=[(256-21)·64+2] ·(1 μs)=15042 μs=15,042 ms ≅ 15 ms

Configuración

Para este caso, la configuración de OPTION_REG sería:

Bits 7 y 6: se usan para configurar PORTB, no usado en este caso, luego da igual su valor: xx

Bit 5: T0CS=0. Usa el reloj interno de instrucción

Bit 4: T0SE=x. Transición usada si CLK externo: no afecta su valor en este caso. Cualquiera

Bit 3: PSA=0. Asigna el prescaler al Timer0.

Bits <2:0> <PS2:PS0> valen 101. Ajusta prescaler del Timer0 a 64.

Por tanto OPTION_REG=b'xx0x0101' Por ejemplo: OPTION_REG=b'00000101'

Uso del Timer en la rutina RETARDO

Uso posterior del Timer0: Se carga el valor 21 (=precarga)

```
          LIST p=16F877A          ; Procesador
          INCLUDE <p16f877A.inc>; Incluye definiciones para procesador
                                  ; Ajusta la palabra de configuración
          __CONFIG    _XT_OSC & _WDT_OFF & _PWRTE_ON & _BODEN_ON & _LVP_OFF

DISPLAY0    EQU 0x20              ; Posición para código para el Display0 en 7 segmentos
DISPLAY1    EQU 0x21              ; Posición para código para el Display1 en 7 segmentos

V_RESET     ORG 0x00              ; Vector de RESET
            goto 0x05             ; Salta el vector de interrupción en 0x04
                                  ; Aunque no utilicemos interrupciones, desactivadas tras el RESET
            ORG 0x05
INICIO      movlw b'00000000'     ; Valores iniciales para PORTA y PORTB
            movwf PORTA           ; Desactiva todos los displays
            movwf PORTD           ; Código inicial en PORTD todos los LED apagados
            bsf STATUS,RP0        ; Banco 1
            movlw b'00000000'     ; Puerto A todas las líneas digitales
            movwf ADCON1
            movlw b'11001111'     ; Lineas RA0 y RA1 salidas.
            movwf TRISA
            movlw b'11111111'     ; Puerto D todo salidas
            clrf TRISD
            movlw b'11001110'     ; Inicializa Timer0
            movwf OPTION_REG
            bsf STATUS,RP1        ; Banco 0
                                  ; Inicializa valores a sacar por los displays
                                  ; DISPLAY0 y DISPLAY1 contienen los valores a volcar sobre PORTD
BARRIDO     movf  DISPLAY0, W     ; Carga el valor a representar en 7 segmentos del Display 0
            movwf PORTD           ; Saca el valor del primer digito por PORTD
            bcf PORTA,0           ; Luego activamos el display 0 (display de decenas)
                                  ; activando la salida correspondiente RA0
            call RETARDO          ; Retardo mientras está activo el display 0
            bsf PORTA,0           ; Desactiva el display 0
            movf  DISPLAY1, W     ; Carga el valor a representar en 7 segmentos del Display 1
            movwf PORTD           ; Saca el valor del primer digito por PORTD
            bcf PORTA,1           ; Luego activamos el display 0 (display de decenas)
                                  ; activando la salida correspondiente RA1
            call RETARDO          ; Retardo mientras está activo el display 0
            bsf PORTA,1           ; Desactiva el display 0
            goto BARRIDO          ; Vuelve a empezar otro barrido

; RETARDO. Genera un retardo durante el cual el display está encendido
RETARDO     bcf INTCON,T0IF     ; Ponemos a cero el flag de overflow del Timer0
            movlw d'178'        ; Valor de precarga para conseguir el tiempo previsto
            movwf TMR0          ; Precarga TMR0 iniciando la temporización
ESPERA      btfss INTCON,T0IF   ; Si rebosa el temporizador, salimos
            goto ESPERA         ; Si no rebosa sigue esperando
            return
            END
```

movlw b‘11111111’
;Con la conexión actual para apagar los displays al pasar a ser salidas debe ser escribir unos en ambos puertos o al menos en uno de ellos

movlw b‘00000110’
Para que las líneas de PORTA sean todas digitales, se ha de escribir ‘xxxxx011x’ en ADCON1. Por tanto, son válidos otros valores, este p.e.

movlw b‘11111100’ ;Líneas RA1 y RA0 de PORTA salidas, resto entradas por seguridad: Válido: ‘xxxxxx00’

Todas las líneas de PORTD salidas. Esto vale ya que borra TRISD.
También serviría movlw b‘00000000’ y movwf TRISD

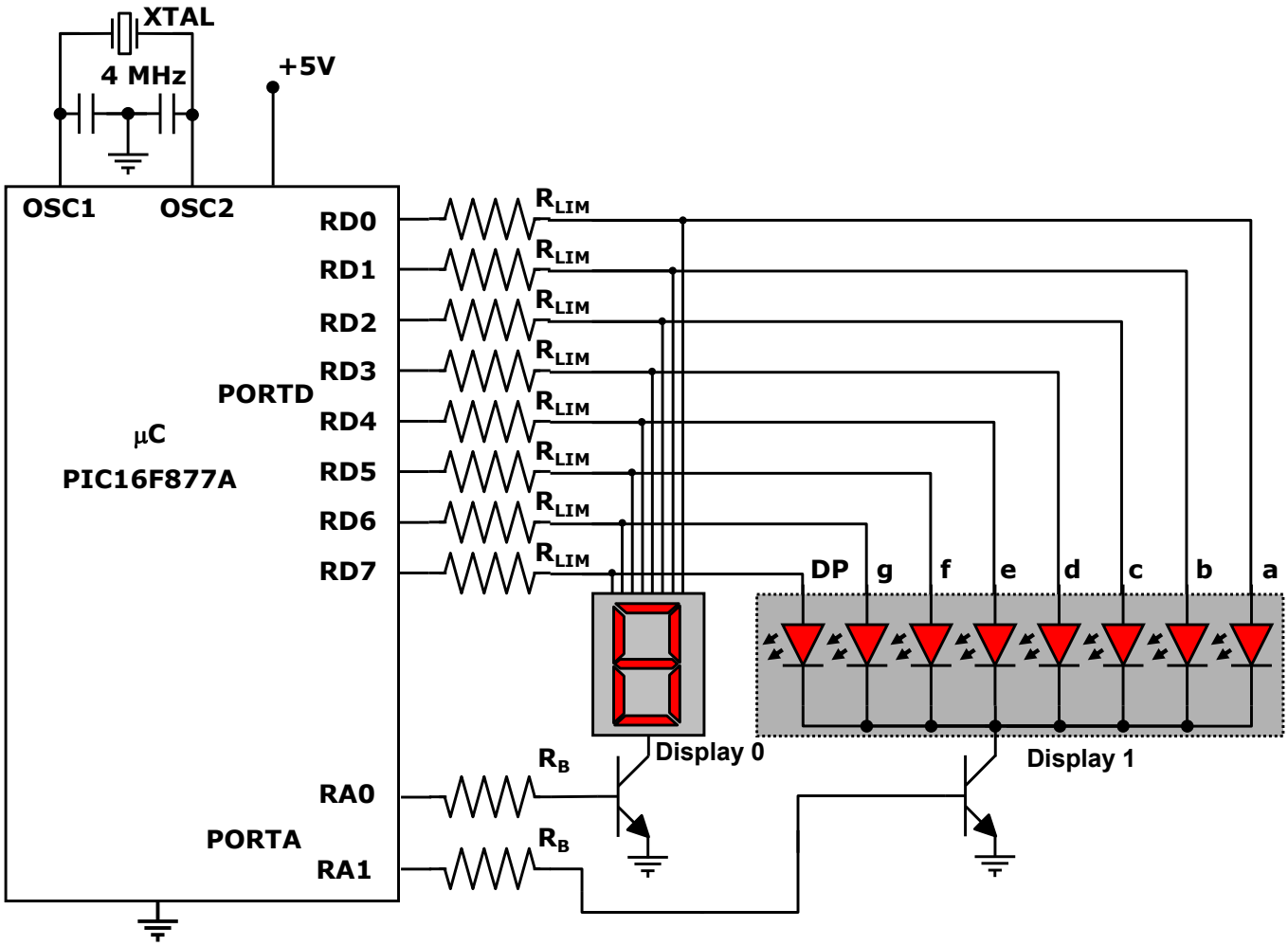
La inicialización del TIMER0 está mal. Ver detalle en cálculos aparte

Está mal. Con esto se pasaría al Banco 3 <RP1 RP0> es <1 1>
La instrucción correcta para ir al Banco 0 sería bcf STATUS, RP0

El valor cargado en TMR0, con la configuración proporcionada, no es correcto para la temporización pedida. Ver detalle en cálculos aparte

EJERCICIO 2.

Se quiere realizar un circuito para realizar el encendido de dos displays desde un microcontrolador PIC16F877A, utilizando el procedimiento de barrido secuencial, para lo que se ha dispuesto el circuito de la figura.



El código a representar en cada display está guardado en las posiciones 0x20 (display0) y 0x21 (display1) (hay otra parte del código, que no se muestra aquí, que se encarga de esta tarea)

La información del valor a mostrar en cada display está codificada directamente en 7 segmentos por lo que el programa a realizar solo debe realizar el encendido de los displays mediante las líneas del PORTA seleccionadas y el volcado de la información de cada display sobre el puerto D. Para cada display, el proceso será

a) Apagar el display
b) Volcar el contenido de la información a mostrar sobre el puerto D
c) Encender el display
d) Esperar un tiempo RETARDO con el display encendido
e) Apagar el display
f) Pasar al siguiente display, empezando el proceso de nuevo en a)

Para conseguir el funcionamiento deseado, nos han proporcionado el siguiente programa:

```
          LIST p=16F877A          ; Procesador
          INCLUDE <p16f877A.inc>; Incluye definiciones para procesador
                                  ; Ajusta la palabra de configuración
          __CONFIG    _XT_OSC & _WDT_OFF & _PWRTE_ON & _BODEN_ON & _LVP_OFF

DISPLAY0    EQU 0x20              ; Posición para código para el Display0 en 7 segmentos
DISPLAY1    EQU 0x21              ; Posición para código para el Display1 en 7 segmentos

V_RESET     ORG 0x00              ; Vector de RESET
            goto 0x05             ; Salta el vector de interrupción en 0x04
                                  ; Aunque no utilicemos interrupciones, desactivadas tras el RESET
            ORG 0x05
INICIO      movlw b'11111111'     ; Valores iniciales para PORTA y PORTB
            movwf PORTA           ; Desactiva todos los displays
            movwf PORTD           ; Código inicial en PORTD todos los LED apagados
            bsf STATUS,RP0        ; Banco 1
            movlw b'00000000'     ; Puerto A todas las líneas digitales
            movwf ADCON1
            movlw b'11001111'     ; Lineas RA0 y RA1 salidas.
            movwf TRISA
            movlw b'11111111'     ; Puerto D todo salidas
            clrf TRISD
            movlw b'11001110'     ; Inicializa Timer0
            movwf OPTION_REG
            bsf STATUS,RP1        ; Banco 0
                                  ; Inicializa valores a sacar por los displays
                                  ; DISPLAY0 y DISPLAY1 contienen los valores a volcar sobre PORTD
BARRIDO     movf  DISPLAY0, W     ; Carga el valor a representar en 7 segmentos del Display 0
            movwf PORTD           ; Saca el valor del primer digito por PORTD
            bsf PORTA,0           ; Luego activamos el display 0 (display de decenas)
                                  ; activando la salida correspondiente RA0
            call RETARDO          ; Retardo mientras está activo el display 0
            bcf PORTA,0           ; Desactiva el display 0
            movf  DISPLAY1, W     ; Carga el valor a representar en 7 segmentos del Display 1
            movwf PORTD           ; Saca el valor del primer digito por PORTD
            bsf PORTA,1           ; Luego activamos el display 0 (display de decenas)
                                  ; activando la salida correspondiente RA1
            call RETARDO          ; Retardo mientras está activo el display 0
            bcf PORTA,1           ; Desactiva el display 0
            goto BARRIDO          ; Vuelve a empezar otro barrido

; RETARDO. Genera un retardo durante el cual el display está encendido
RETARDO     bcf INTCON,T0IF     ; Ponemos a cero el flag de overflow del Timer0
            movlw d'178'        ; Valor de precarga para conseguir el tiempo previsto
            movwf TMR0          ; Precarga TMR0 iniciando la temporización
ESPERA      btfss INTCON,T0IF   ; Si rebosa el temporizador, salimos
            goto ESPERA         ; Si no rebosa sigue esperando
            return
            END
```

DATOS:

a) El cristal del oscilador para el ciclo de instrucción interno tiene una frecuencia $f_{CLK\text{-}INT}$=4 MHz

b) El tiempo de encendido de cada display se obtiene mediante la subrutina RETARDO y se desea que sea de 10 ms.

Se pide:

a) Calcular, con la actual configuración del temporizador, y la actual subrutina de temporización, cuál sería el tiempo que se temporiza realmente en la subrutina "RETARDO", justificando la respuesta

b) Si el tiempo no es correcto, proponer los cambios a realizar en el programa (tanto en la configuración como en el uso del TIMER TMR0) para conseguir temporizar el tiempo solicitado, justificando la respuesta.

c) Con la conexión del PIC que se indica, analizar si la configuración y el uso posterior de los puertos utilizados es correcta para conseguir el propósito del programa. Justificar si el programa es correcto y, de no ser así, indicar los errores cometidos y los cambios a realizar en el programa para conseguir un correcto funcionamiento, coherente con el esquema eléctrico del circuito, justificando la respuesta.

SOLUCIÓN EJERCICIO 2: MODELO CON DOS DISPLAYs CÁTODO COMÚN

PORTD usado para selección de display mediante NPNs a los cátodos comunes (1 enciende, 0 apaga), PORTA saca el código en siete segmentos (1 enciende, 0 apaga)

a) Configuración actual:

Inicialización: OPTION_REG = b '11001110'. Los bits 7 y 6 no nos importan

Bit 5: T0CS=0. Usa el reloj interno de instrucción

Bit 4: T0SE=0. Transición usada si CLK externo: no afecta su valor en este caso

Bit 3: PSA=1. Asigna el prescaler al Watchdog. Por tanto el Timer0 no usa prescaler, y la cuenta es directa (es como si fuera prescaler=1)

Bits <2:0> <PS2:PS0> valen 110. El prescaler del Watchdog es 64, pero no nos afecta su valor para la temporización del Timer0.

Uso del Timer0: Se carga el valor 178 (=precarga)

El tiempo temporizado es entonces:

Tiempo=[(256-precarga)·Prescaler+2]·$T_{INSTRUCCIÓN}$

$T_{INSTRUCCIÓN}=4\cdot T_{CLK}=4/f_{CLK}=4/4Mhz=1\ \mu s$

Tiempo=[(256-178)·1+2]·(1 μs)=80 μs

b) Queremos temporizar 10 ms

En primer lugar, calculamos el máximo valor que se podría temporizar con el reloj interno (único disponible). Para eso ponemos el prescaler al máximo (Prescaler=256) y el valor cargado en TMR0 al menor valor posible, para maximizar la expresión:

Tiempo=[(256-precarga)·Prescaler+2]·$T_{INSTRUCCIÓN}$

$T_{INSTRUCCIÓN}=4\cdot T_{CLK}=4/f_{CLK}=4/4Mhz=1\ \mu s$

El máximo tiempo que se puede temporizar en una sola temporización es:

Tiempo=[(256-0)·256+2] ·(1 μs)=65538 μs=64,538 ms

Podemos, por tanto, temporizar sin problema los 15 ms. Para conseguirlo:

Tiempo=[(256-precarga)·Prescaler+2]·$T_{INSTRUCCIÓN}$ $\Rightarrow$

$$[(256-precarga)\cdot Prescaler+2=\frac{Tiempo}{T_{INSTRUCCIÓN}}\Rightarrow$$

$$[(256 - precarga) \cdot Prescaler = \frac{Tiempo}{T_{INSTRUCCIÓN}} - 2 \Longrightarrow [(256 - precarga) = \frac{\frac{Tiempo}{T_{INSTRUCCIÓN}} - 2}{Prescaler} \Longrightarrow$$

$$precarga = 256 - \frac{\frac{Tiempo}{T_{INSTRUCCIÓN}} - 2}{Prescaler} = 256 - \frac{\frac{10ms}{1\mu s} - 2}{Prescaler} = 256 - \frac{10000 - 2}{Prescaler} = 256 - \frac{9998}{Prescaler}$$

Valores posibles:

Prescaler	Precarga	Conclusión
256	216,94≅217	Válido
128	177,89≅178	Válido
64	99,78≅100	Válido
32	-56,43 <0	Este valor de prescaler ya no sirve

Según el valor escogido, se ajusta OPTION_REG y el valor a cargar en el TMR0 en la rutina RETARDO

Por ejemplo, si tomamos Prescaler=64, entonces en TMR0 precargamos 100 en la rutina RETARDO. Comprobamos que la temporización sería:

Tiempo=[(256-100)·64+2] ·(1 μs)=9986 μs=9,986 ms ≅ 10 ms

Configuración

Para este caso, la configuración de OPTION_REG sería:

Bits 7 y 6: se usan para configurar PORTB, no usado en este caso, luego da igual su valor: xx

Bit 5: T0CS=0. Usa el reloj interno de instrucción

Bit 4: T0SE=x. Transición usada si CLK externo: no afecta su valor en este caso. Cualquiera

Bit 3: PSA=0. Asigna el prescaler al Timer0.

Bits <2:0> <PS2:PS0> valen 101. Ajusta prescaler del Timer0 a 64.

Por tanto OPTION_REG=b'xx0x0101' Por ejemplo: OPTION_REG=b'00000101'

Uso del Timer en la rutina RETARDO

Uso posterior del Timer0: Se carga el valor 100 (=precarga)

```
          LIST p=16F877A          ; Procesador
          INCLUDE <p16f877A.inc>; Incluye definiciones para procesador
                                  ; Ajusta la palabra de configuración
          __CONFIG    _XT_OSC & _WDT_OFF & _PWRTE_ON & _BODEN_ON & _LVP_OFF

DISPLAY0    EQU 0x20              ; Posición para código para el Display0 en 7 segmentos
DISPLAY1    EQU 0x21              ; Posición para código para el Display1 en 7 segmentos

V_RESET     ORG 0x00              ; Vector de RESET
            goto 0x05             ; Salta el vector de interrupción en 0x04
                                  ; Aunque no utilicemos interrupciones, desactivadas tras el RESET
            ORG 0x05
INICIO      movlw b'11111111'     ; Valores iniciales para PORTA y PORTB
            movwf PORTA           ; Desactiva todos los displays
            movwf PORTD           ; Código inicial en PORTD todos los LED apagados
            bsf STATUS,RP0        ; Banco 1
            movlw b'00000000'     ; Puerto A todas las líneas digitales
            movwf ADCON1
            movlw b'11001111'     ; Lineas RA0 y RA1 salidas.
            movwf TRISA
            movlw b'11111111'     ; Puerto D todo salidas
            clrf TRISD
            movlw b'11001110'     ; Inicializa Timer0
            movwf OPTION_REG
            bsf STATUS,RP1        ; Banco 0
                                  ; Inicializa valores a sacar por los displays
                                  ; DISPLAY0 y DISPLAY1 contienen los valores a volcar sobre PORTD
BARRIDO     movf  DISPLAY0, W     ; Carga el valor a representar en 7 segmentos del Display 0
            movwf PORTD           ; Saca el valor del primer digito por PORTD
            bsf PORTA,0           ; Luego activamos el display 0 (display de decenas)
                                  ; activando la salida correspondiente RA0
            call RETARDO          ; Retardo mientras está activo el display 0
            bcf PORTA,0           ; Desactiva el display 0
            movf  DISPLAY1, W     ; Carga el valor a representar en 7 segmentos del Display 1
            movwf PORTD           ; Saca el valor del primer digito por PORTD
            bsf PORTA,1           ; Luego activamos el display 0 (display de decenas)
                                  ; activando la salida correspondiente RA1
            call RETARDO          ; Retardo mientras está activo el display 0
            bcf PORTA,1           ; Desactiva el display 0
            goto BARRIDO          ; Vuelve a empezar otro barrido

; RETARDO. Genera un retardo durante el cual el display está encendido
RETARDO     bcf INTCON,T0IF     ; Ponemos a cero el flag de overflow del Timer0
            movlw d'178'        ; Valor de precarga para conseguir el tiempo previsto
            movwf TMR0          ; Precarga TMR0 iniciando la temporización
ESPERA      btfss INTCON,T0IF   ; Si rebosa el temporizador, salimos
            goto ESPERA         ; Si no rebosa sigue esperando
            return
            END
```

movlw b‘00000000’
;Con la conexión actual para apagar los displays al pasar a ser salidas debe ser escribir ceros en ambos puertos o al menos en uno de ellos

movlw b‘00000110’
Para que las líneas de PORTA sean todas digitales, se ha de escribir ‘xxxxx011x’ en ADCON1. Por tanto, son válidos otros valores, este p.e.

movlw b‘11111100’ ;Líneas RA1 y RA0 de PORTA salidas, resto entradas por seguridad: Válido: ‘xxxxxx00’

Todas las líneas de PORTD salidas. Esto vale ya que borra TRISD.
También serviría movlw b‘00000000’ y movwf TRISD

La inicialización del TIMER0 está mal. Ver detalle en cálculos aparte

Está mal. Con esto se pasaría al Banco 3 <RP1 RP0> es <1 1>
La instrucción correcta para ir al Banco 0 sería bcf STATUS, RP0

El valor cargado en TMR0, con la configuración proporcionada, no es correcto para la temporización pedida. Ver detalle en cálculos aparte