Diseño de un módulo para control de 8 servos y 5 canales analógicos por bus I^2C

Alejandro Alonso Puig – mundobot.com
Octubre 2003

Introducción 2
Diseño Electrónico 2
Diseño del software 5
Registros 5
Comandos I^2C 6
Escritura registro 7
Lectura Registro 7
Cuerpo del programa 8
Código fuente 8
Ejemplo de Master 21

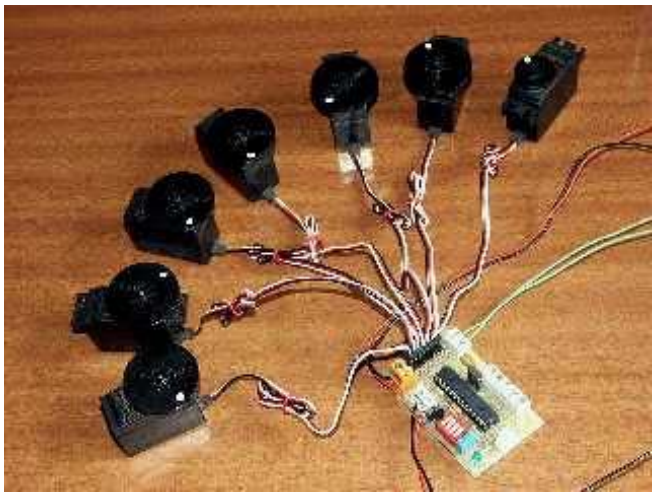
Introducción

El siguiente informe técnico describe el diseño, tanto desde el punto de vista electrónico, como informático de un placa para control por bus I^2C de 8 servos estándar de radiocontrol y 5 entradas analógicas (Conversión analógica/digital).

Las características principales del módulo presentado son las siguientes:

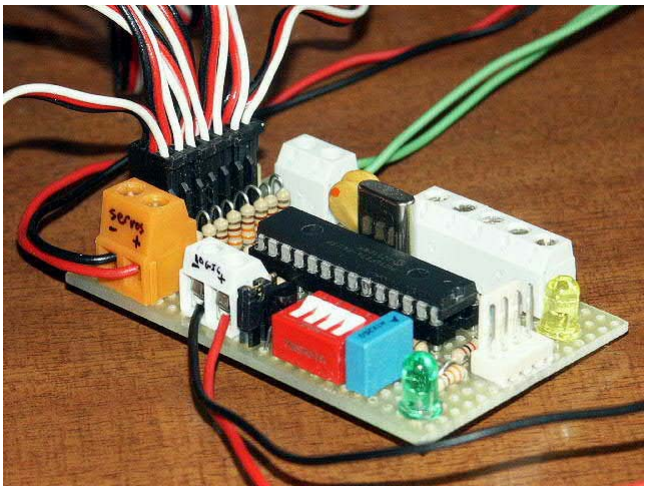
- Actúa como Slave permitiendo seleccionar mediante switches dip la dirección que utilizará en la red I^2C.
- Se puede establecer mediante bus I^2C tanto la posición deseada de cada servo, como el sentido de giro y el *Offset*.
- Se puede leer por bus I^2C el valor digital (8 bits) correspondiente a cualquiera de las 5 entradas analógicas de las que dispone..

La ventaja que se obtiene con este tipo de módulos es precisamente el control por bus I^2C que permite tener varios módulos de este tipo conectados al mismo bus. De esta manera pueden controlarse gran cantidad de servos desde un controlador principal sin apenas sobrecarga en el mismo. Adicionalmente se tiene medida de valores analógicos, muy útil para determinado tipo de sensores.



Diseño Electrónico

El módulo está compuesto básicamente por en un microcontrolador PIC 16F876 de Microchip funcionando a 4Mhz.



Este microcontrolador posee un módulo MSSP que permite el control de las comunicaciones I^2C por Hardware. Se utilizan los pines SCL (RC3) y SDA (RC4) conectados a resistencias Pull-Up de 10KΩ para las comunicaciones I^2C.

Adicionalmente posee un grupo de cinco conversores analógico digitales (A/D) que pueden soportar entradas analógicas entre 0 y 5 voltios. Los valores analógicos serán relativos a la masa del circuito, por lo que la masa del elemento cuya señal analógica se quiera medir deberá estar conectada a la masa del modulo que presentamos.

Existen 4 entradas (RB2-RB5) conectadas a un switch-dip cuádruple. Solo las dos inferiores (RB2, RB3) son utilizadas por el firmware actual

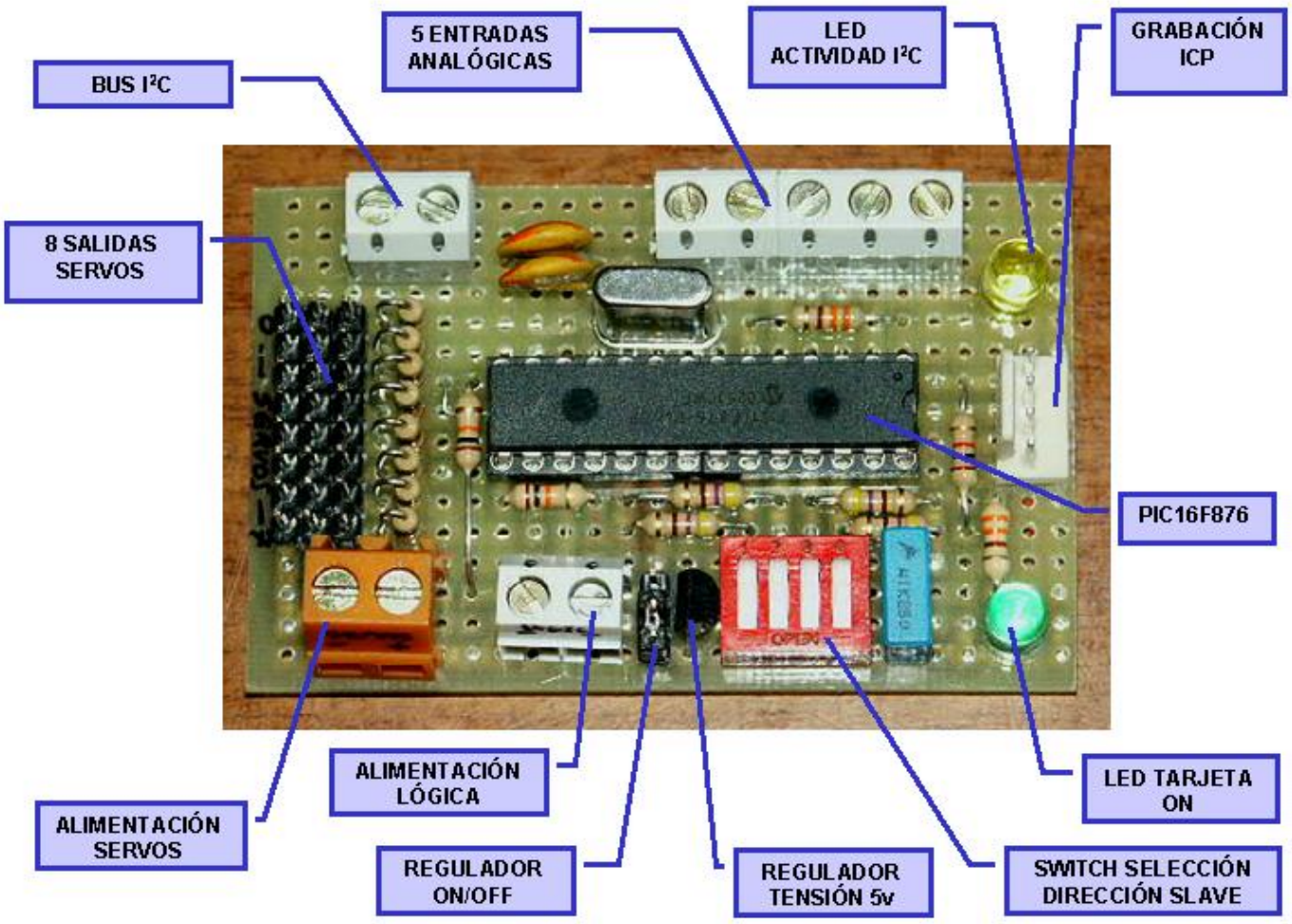
y permitirán seleccionar diferentes direcciones I^2C para evitar coincidencia de direccionamiento con otros dispositivos (entre 01110000 [0x70] y 01110110 [0x76]).

Tiene doble entrada de alimentación, una para los servos y otra para la lógica. Utilizar doble alimentación no es obligatorio, aunque si es recomendable para evitar interferencias en el funcionamiento de la lógica. En cualquier caso el módulo incluye un regulador de tensión a 5 voltios (7805) que puede ser activado o desactivado mediante jumper, según se vaya a alimentar el circuito con 5 voltios o más. Obviamente la masa de ambas tomas de alimentación es común.

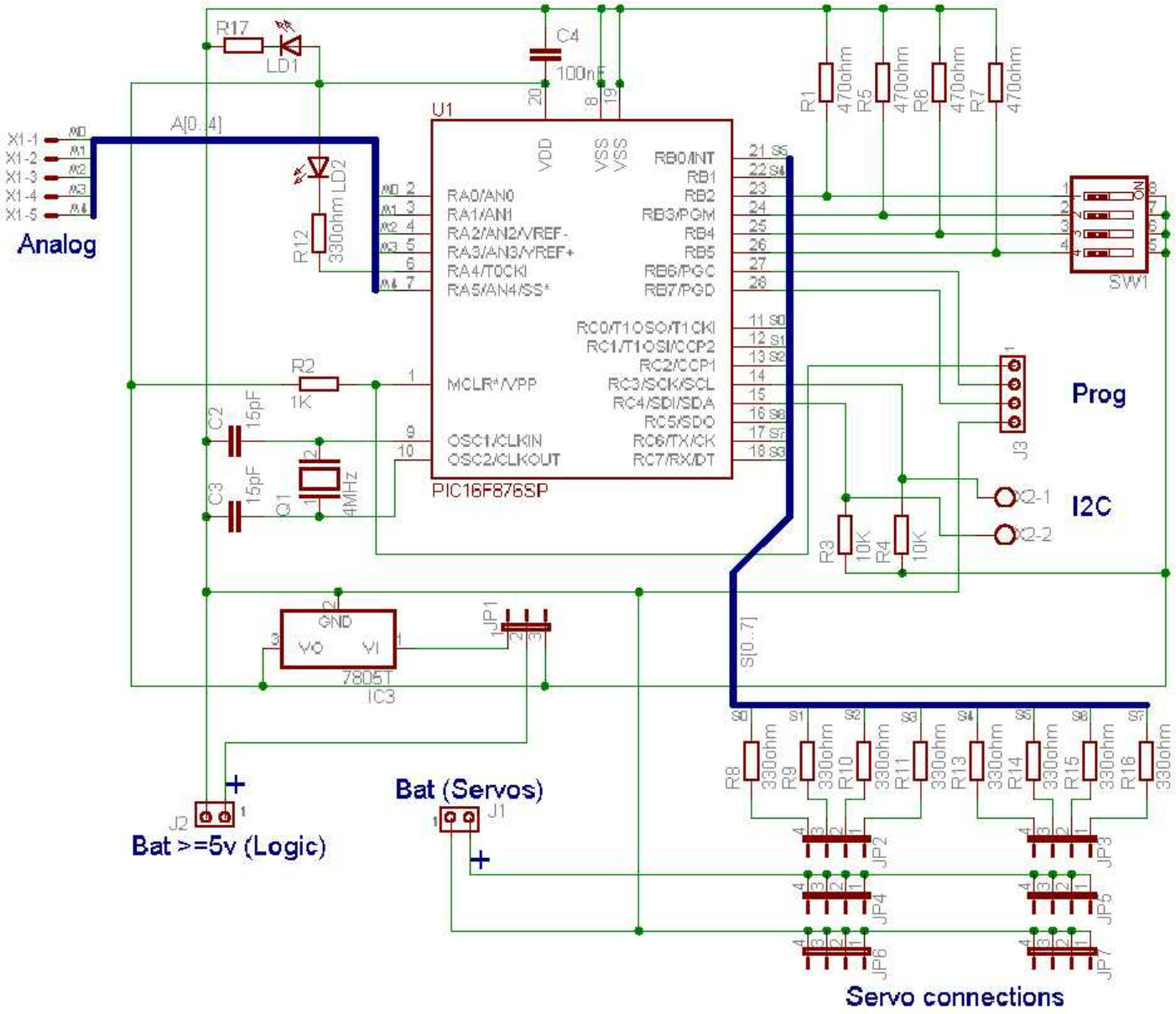
Finalmente los pines MCLR, RB6 y RB7 se han llevado junto con la masa a un conector cuádruple con el fin de permitir la programación del chip una vez montado en placa (Programación ICP).

Las salidas para el control del los servos incluyen resistencias de 330Ω para prevenir daños en el microcontrolador en caso de exceso de consumo en el circuito interno de los servos.

El circuito no está protegido en caso de conectar la alimentación incorrectamente, por lo que se ha de prestar atención a la polaridad.



El esquema electrónico completo se muestra a continuación.



Diseño del software

El software ha sido desarrollado en ensamblador para PIC16F876.

El módulo está configurado para producir una interrupción cada vez que se desborda el timer TMR0 o cada vez que se produce un evento I^2C. Al atender dicha interrupción verifica cual de los dos tipos es y actúa en consecuencia:

- ✓ El TMR0 se configura con un valor adecuado para que se desborde cada 20msg, que es el tiempo en el que se ha de refrescar el valor PWM enviado a los servos. Cuando se produce una interrupción de este tipo, se atienden los 8 servos secuencialmente, vigilando asimismo si se produce simultáneamente un evento I^2C. En caso de producirse, se forzará la parada de la señal de reloj I^2C para que el Master no envíe más información hasta que se tenga tiempo de atender el evento que ha llegado. Así se impiden overflows.

- ✓ Si el evento que ha producido la interrupción es I^2C, se atenderá en la rutina "SSP_Handler", que verificará la adecuada llegada de tramas y se ocupará de actualizar o leer una serie de registros de los que hablaremos más adelante y que serán los responsables del funcionamiento del módulo.

Registros

El módulo contiene una serie de registros que pueden ser accedidos desde un Master por I^2C. Algunos de estos registros son solo de lectura, mientras que otros son de escritura.

Mediante el manejo de estos registros se obtiene toda la funcionalidad del módulo. A continuación describimos dichos registros:

- **Modo:** Este registro permite al Master especificar el modo de funcionamiento de cada servo: Normal (giro normal) y Reverse (sentido inverso)

  Cada bit representa un servo. Así por ejemplo, si Modo es 10010100, los servos 0, 1, 3, 5 y 6 girarán en sentido normal y los servos 2, 4 y 7 funcionarán en sentido inverso.

  Por defecto este registro vale 0, con lo que todos los servos giran en sentido normal.

- **Offset**: Es un registro de solo escritura cuyo valor por defecto es 70. Permite ajustar la posición cero del servo. Es decir, debido a que cada modelo de servo tiene unas características diferentes, la posición cero de dos servos diferentes puede darse en diferentes *Duty Cycle*s (Anchura del pulso positivo). Por ejemplo en un servo se puede dar con un *Duty Cycle* de 1msg, mientras que en otro se puede dar con un *Duty Cycle* de 400µsg. Por ello ha de ajustarse el *Offset* a las características del servo. Ha de tenerse en cuenta que una variación de una unidad en el *Offset* implica una variación de 5µsg en el *Duty Cycle*. Asimismo ha de tenerse en cuenta que el valor del *Duty Cycle* vendrá dado en µsg por la siguiente fórmula en este módulo:

$$Duty\ Cycle = 10 + 5x Offset + 8x Posic$$

De esta manera, para la posición 0, el *Offset* será:

$$Offset = \frac{DutyCycle - 10}{5}$$

Así, para un *Duty Cycle* de 1msg (1000μsg) en la posición 0, el *Offset* habrá de ser 198

- **Posic**: Es un registro de solo escritura cuyo valor por defecto es 0. Indica la posición que se desea que tenga el servo. El límite depende del modelo de servo. Básicamente ha de tenerse en cuenta que un incremento de una unidad en este valor implica un incremento de 8 μsg en el *Duty Cycle*. Según el valor del *Offset* de un servo y del valor de *Posic*, se tendrá un *Duty Cycle* concreto según la fórmula que se mostró antes. Por ejemplo para un *Offset* de valor 70 (valor por defecto) y un valor de *Posic* de 220 generaría una señal PWM de *Duty Cycle* = 2.120μsg.

  Por tanto se tiene que los valores límites de *Duty Cycle* que se pueden conseguir con este módulo varían desde 10μsg (*Offset*=0, *Posic*=0) a 3.325μsg (*Offset*=255, *Posic*=255)

  Los valores son muy extremos, por lo que hay que tener cuidado en su asignación para no dañar el servo. En caso de no conocerse los límites de *Duty Cycle* de un servo puede procederse de la siguiente manera: Establecer para un servo concreto desde el Master un valor de *Posic*=0 y un *Offset*=198. Esto generará un PWM de *Duty Cycle* = 1000μsg, valor aceptado por cualquier servo comercial. Si se observa que el eje del servo está lejos de su posición de límite físico, se irá reduciendo el valor del *Duty Cycle* hasta que el eje esté cerca del límite físico. No ha de llegar en ningún momento al límite físico ya que podría dañar el mecanismo de engranajes del servo. Una vez establecido el valor de *Offset*, se puede ir incrementando progresivamente el valor de *Posic* hasta que el eje casi llegue a su otro límite físico. Con esto ya se ha conseguido saber el valor que ha de tener *Offset* y los valores límites de *Posic*

- **Analog**: Es un registro de solo lectura. Indica el valor digital de la conversión A/D de una entrada analógica concreta. La resolución usada en la conversión A/D es de 8 bits, es decir que lo mínimo detectable son variaciones de 19,5mV.

- **Versión**: Indica la versión del firmware. Es un registro de solo lectura.

Comandos I^2C

Básicamente las órdenes que llegarán del Master al Slave serán de lectura y/o escritura en los registros mencionados anteriormente.

En la primera trama se indica el número de registro al que se quiere acceder, bien sea para lectura o escritura. Dicha numeración es la siguiente:

1. Registro de Modo
2. Offset servo
3. Posición servo
4. Conversor A/D
5. No usado
6. No usado
7. Revisión Firmware.

En la segunda trama se indica el número de dispositivo al que nos referimos (número de servo o número de entrada analógica)

Si estamos escribiendo un registro, en la tercera trama irá el valor que escribimos. Si por el contrario estamos accediendo en modo lectura, se enviará orden de solicitud del valor del registro.

Por ejemplo, si se quiere ordenar que el servo número 0 tenga un *Offset* de 92, habría que enviar un bloque con cuatro bytes:

- ✓ Dirección Slave en modo escritura (si no se cambian los switches será 112 (01110000))
- ✓ Nº Registro a acceder (*Offset*=2)
- ✓ Nº Dispositivo a acceder (Servo0 --> 0)
- ✓ Valor Registro: 92

Si lo que se quiere por ejemplo es leer el valor de la entrada analógica número 3, se enviaría un primer bloque con 3 bytes:

- ✓ Dirección Slave en modo escritura (si no se cambian los switches será 112 (01110000))
- ✓ Nº Registro a acceder (*A/D*=4)
- ✓ Nº Dispositivo a acceder (Entrada nº 3 --> 3)

Tras esto se enviaría un segundo bloque con un byte:

- ✓ Dirección Slave en modo lectura (si no se cambian los switches será 113 (01110001))

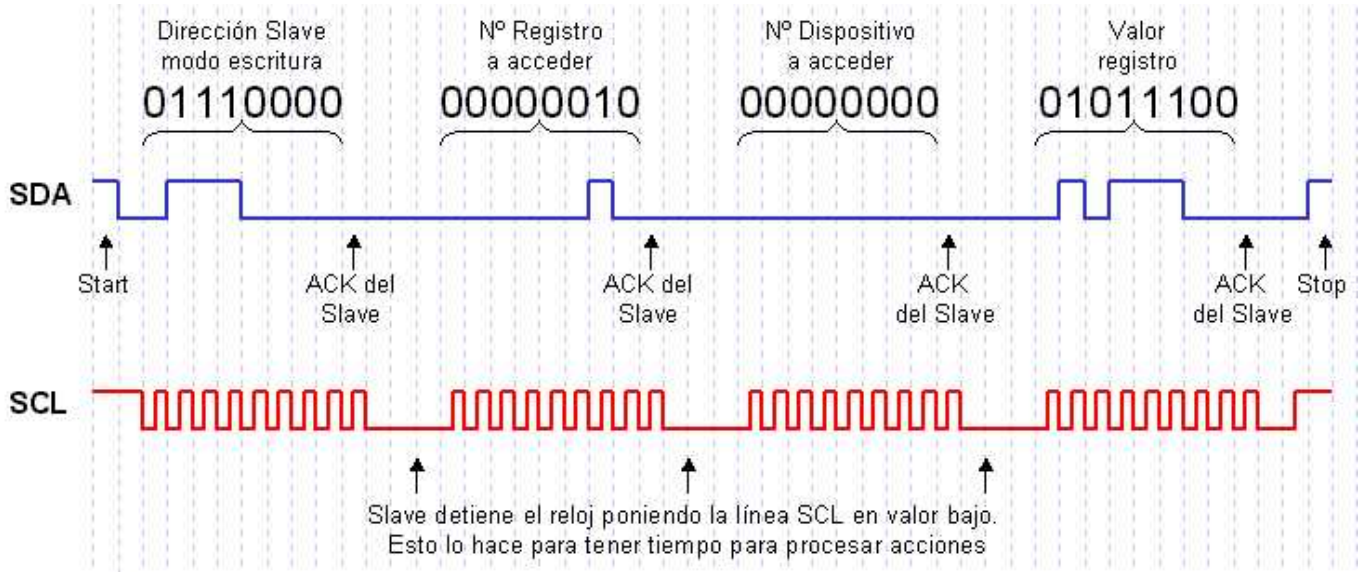
.. y se recibirá del Slave un byte correspondiente al dato solicitado.

A continuación se muestra el formato de los comandos I^2C utilizados para el intercambio de datos entre Master y Slave a nivel de lógica digital.

Escritura registro

En caso de querer escribir en un registro, se enviará desde el Master al Slave un solo bloque de datos con cuatro bytes (Dirección Slave en modo escritura, nº registro, nº dispositivo, valor registro).

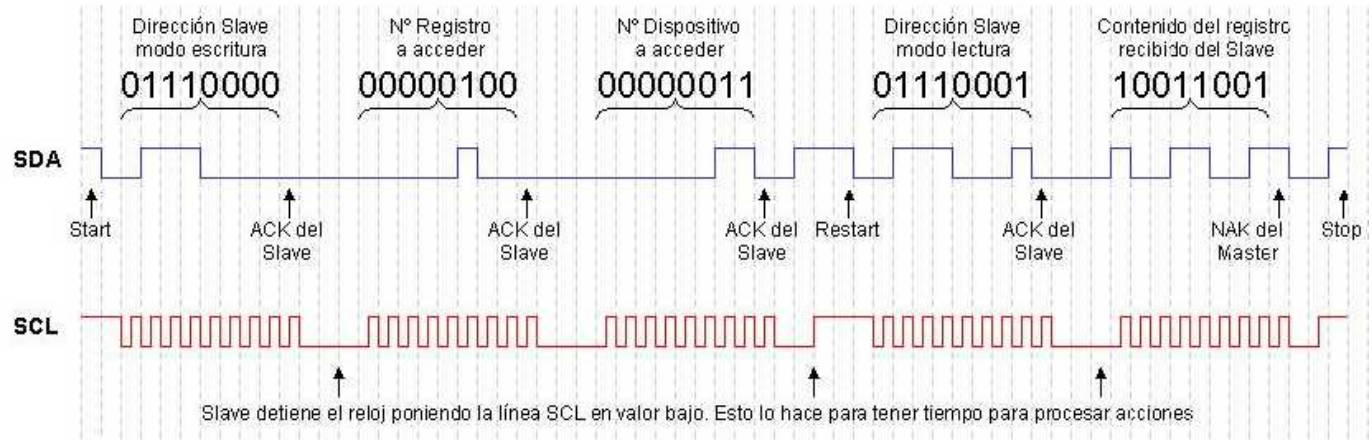
Tomando el ejemplo anterior de establecimiento de un valor 92 para el *Offset*, la representación de la transmisión a nivel de lógica digital sería como sigue:



Lectura Registro

En caso de querer leer de un registro, se enviará desde el Master al Slave un bloque con tres bytes (Dirección Slave en modo escritura, nº registro, nº dispositivo) y otro bloque con un byte (Dirección Slave en modo lectura). El Slave contestará con otro bloque de un byte (valor de registro solicitado)

Tomando el ejemplo anterior de lectura de un valor del conversor A/D, la representación de la transmisión a nivel de lógica digital sería como sigue:



Cuerpo del programa

Básicamente el funcionamiento del programa consiste en un bucle en el que mientras no se produzca una interrupción por evento I^2C o por desbordamiento del TMR0, Va haciendo conversiones A/D de sus 5 entradas analógicas y dejándolas en variables para que en caso de ser solicitadas por I^2C pueda entregar los valores de forma inmediata. Esto se hace porque la toma de un valor analógico y su conversión a un dato digital toma cierto tiempo. Teniendo siempre los valores ya convertidos, se agiliza la transmisión.

Código fuente

A continuación se incluye el código fuente del módulo comentado. Incluye comentarios que aclararán su funcionamiento.

```
; SvI2C02
; Programa para control de servos y entradas analógicas por bus I2C
; Por: Alejandro Alonso Puig
; Fecha: 11/10/2003
; Controlador: 16F876 4Mhz
; Función:
; Control de hasta 8 servos y 5 entradas analógicas.
; Dispone de las siguientes características:
; -Valores de posicionamiento (0 a 255)
; -Permite funcionar a cada servo en modo normal o reverse
; -Permite establecer un Offset por servo para fijar posición inicial
; -Switches dip para especificar la dirección i2c que se utilizará
; -Conversión A/D de 5 entradas analógicas
;
```

```
; La anchura de flanco alto (Duty Cycle) enviada a cada servo vendrá dada en usg por la formula
; Duty Cycle en usg = 10 + 5xOffset + 8xPosic en caso de funcionamiento normal y
; Duty Cycle en usg = 10 + 5xOffset + 8x(255-Posic) en caso de funcionamiento reverse



        list            p=16F876
        include         "P16F876.INC"

;Definición de macros


        #define Version         d'02'   ;Versión del programa

        ;Puertos
        #define Servo0  PORTC,0         ; Servo nº0
        #define Servo1  PORTC,1         ; Servo nº1
        #define Servo2  PORTC,2         ; Servo nº2
        #define Servo3  PORTC,7         ; Servo nº3
        #define Servo4  PORTB,1         ; Servo nº4
        #define Servo5  PORTB,0         ; Servo nº5
        #define Servo6  PORTC,5         ; Servo nº6
        #define Servo7  PORTC,6         ; Servo nº7

        #define S0      .0              ;Referencia al servo 0 en palabras de estado
        #define S1      .1              ;Referencia al servo 1 en palabras de estado
        #define S2      .2              ;Referencia al servo 2 en palabras de estado
        #define S3      .3              ;Referencia al servo 3 en palabras de estado
        #define S4      .4              ;Referencia al servo 4 en palabras de estado
        #define S5      .5              ;Referencia al servo 5 en palabras de estado
        #define S6      .6              ;Referencia al servo 6 en palabras de estado
        #define S7      .7              ;Referencia al servo 7 en palabras de estado

        #define Led     PORTA,4         ;Led de actividad I2C

;Definición de variables

        cblock  0x20

        ;Variables para control de servos
        DispNo          ;Numero de dispositivo accedido desde I2C (Servo, pto Analog)
        Posic           ;Variable Posicion Servo para cálculo retardo
        Posic0          ;Posicion Servo nº0 (0 a 255)
        Posic1          ;Posicion Servo nº1 (0 a 255)
        Posic2          ;Posicion Servo nº2 (0 a 255)
        Posic3          ;Posicion Servo nº3 (0 a 255)
        Posic4          ;Posicion Servo nº4 (0 a 255)
        Posic5          ;Posicion Servo nº5 (0 a 255)
        Posic6          ;Posicion Servo nº6 (0 a 255)
        Posic7          ;Posicion Servo nº7 (0 a 255)

        ;Offsets
        ;El valor de Offset permite establecer la posición inicial de cada en función
        ;de la anchura del pulso activo en microsegundos. El cálculo se hace mediante
        ;la fórmula Offset=(T-10)/5, donde T es la anchura del pulso activo en usg
        ;Se utiliza para dos funciones: Ajustar varios servos para que sus ejes estén
        ;colocados físicamente en la misma posición cero, adaptando los
        ;offsets de cada servo. También se utiliza para aprovechar al máximo el giro
        ;posible del eje, que varia para cada marca y modelo.
        ;Por ejemplo, si queremos establecer que el valor de 0 implique un pulso
        ;activo de una anchura de 1000usg, el offset será (1000-10)/5=198
        Offset0         ;Offset Servo nº0
        Offset1         ;Offset Servo nº1
        Offset2         ;Offset Servo nº2
        Offset3         ;Offset Servo nº3
        Offset4         ;Offset Servo nº4
        Offset5         ;Offset Servo nº5
        Offset6         ;Offset Servo nº6
        Offset7         ;Offset Servo nº7
        Offset          ;Valor Offset servo

        Modo            ;Palabra de estado que indica en cada bit que servo girará a la inversa

        APort           ;Puerto analógico a medir
        APortTmp        ;Variable temporal utilizada en la medición
        Analog          ;Valor analógico medido
```

```
        Analog0         ;Valor analógico puerto 0
        Analog1         ;Valor analógico puerto 1
        Analog2         ;Valor analógico puerto 2
        Analog3         ;Valor analógico puerto 3
        Analog4         ;Valor analógico puerto 4

        PDel0           ;Usada en retardos
        ADel0           ;Usada en retardos
        BDel0           ;Usada en retardos
        BDel1           ;Usada en retardos
        BDel2           ;Usada en retardos
        Pausa           ;Usada en para hacer pausas con subr "HacerTiempo"
        Temp            ;Variable Temporal para usos puntuales en cálculos
        Temp2           ;Variable Temporal para usos puntuales en cálculos
        Temp3           ;Variable Temporal para usos puntuales en cálculos
        BkStatus        ;Backup del registro STATUS
        BkW             ;Backup W
        BkStatus2       ;Backup del registro STATUS (Interrupciones)
        BkW2            ;Backup W (Interrupciones)

        DirNodo         ;Dirección I2C de este slave (4 posibles direcciones segun switches dip)
        MensajeIn       ;Contendrá el dato recibido por I2C del master
        MensajeOut      ;Contendrá el dato a enviar por I2C al master
        Registro        ;Registro accedido por i2c:
                        ;       1: Registro de Modo. Normal o Reverse servo
                        ;       2: Offset servo
                        ;       3: Posición servo
                        ;       4: Valor puerto analógico
                        ;       5: No usado
                        ;       6: No usado
                        ;       7: Revisión Firmware. "Version"
        StatI2C         ;Registro intermedio para identificar el estado i2c:
                        ;       0: Si On --> llegó nº Registro
                        ;       1: Si On --> llegó nº Dispositivo al que se referirá el registro

        endc            ;Fin de definiciones



        org     0
        goto    INICIO
        org     5


;*****************************************************************************
Interrupcion            ;RUTINA DE INTERRUPCIÓN. Activa flancos segun valor de variables
                ;de Posicion (Servos) y trata eventos I2C
;*****************************************************************************



        ;Guardamos copia de algunos registros
        movwf   BkW2            ;Hace copia de W
        movf    STATUS,W        ;Hace copia de registro de estado
        banksel BkStatus2
        movwf   BkStatus2

        ;Chequeamos si la interrupción es por evento I2C. En caso positivo llamamos
        ;a la rutina de proceso del evento
        banksel PIR1
        btfss   PIR1,SSPIF      ;Ha ocurrido un evento SSP? (I2C)
        goto    IntNoSSP        ;No. entonces será por otra cosa. Saltamos.
        bcf     SSPCON,CKP      ;Si. Detenemos reloj I2C para evitar desbordamiento o timeout
        banksel PORTA
        bcf     Led             ;Enciende led de actividad I2C
        call    SSP_Handler     ;Procesamos el evento.
        banksel PIR1
        bcf     PIR1,SSPIF      ;Limpiamos el flag
        goto    Rest


IntNoSSP        ;Aquí se gestionan interrupciones que no son por SSP



        ;Es una interrupción de desbordamiento del TMR0 --> Gestión Servos
```

```
        bcf     INTCON,T0IF     ;Repone flag del TMR0
        movlw   d'177'          ;Repone el TMR0 con 177 (complemento de 78) -182
        banksel TMR0
         movwf  TMR0            ;256*78=19.968 (casi 20.000 usg= 20ms)


        ;** SERVO0 **
        movlw   d'255'          ;Precargamos Posic con 255...
        movwf   Posic           ;...por si ha de funcionar en reverse mode
        movf    Posic0,W        ;Carga variable Posic con valor posición para servo
        btfsc   Modo,S0         ;Verifica si ha de funcionar en reverse mode
        goto    Inv0            ;Si. invertimos valor
        movwf   Posic           ;No
        goto    Sigue0
Inv0    subwf   Posic,F         ;Si. invertimos valor
Sigue0  movf    Offset0,w       ;Carga Offset del servo correspondiente
        movwf   Offset
        bsf      Servo0         ;activamos flanco
        call    Retardo         ;dejamos activo el tiempo necesario
        bcf      Servo0         ;bajamos flanco

        ;** SERVO1 **
        movlw   d'255'          ;Precargamos Posic con 255...
        movwf   Posic           ;...por si ha de funcionar en reverse mode
        movf    Posic1,W        ;Carga variable Posic con valor posición para servo
        btfsc   Modo,S1         ;Verifica si ha de funcionar en reverse mode
        goto    Inv1            ;Si. invertimos valor
        movwf   Posic           ;No
        goto    Sigue1
Inv1    subwf   Posic,F         ;Si. invertimos valor
Sigue1  movf    Offset1,w       ;Carga Offset del servo correspondiente
        movwf   Offset
        bsf      Servo1         ;activamos flanco
        call    Retardo         ;dejamos activo el tiempo necesario
        bcf      Servo1         ;bajamos flanco

        ;** SERVO2 **
        movlw   d'255'          ;Precargamos Posic con 255...
        movwf   Posic           ;...por si ha de funcionar en reverse mode
        movf    Posic2,W        ;Carga variable Posic con valor posición para servo
        btfsc   Modo,S2         ;Verifica si ha de funcionar en reverse mode
        goto    Inv2            ;Si. invertimos valor
        movwf   Posic           ;No
        goto    Sigue2
Inv2    subwf   Posic,F         ;Si. invertimos valor
Sigue2  movf    Offset2,w       ;Carga Offset del servo correspondiente
        movwf   Offset
        bsf      Servo2         ;activamos flanco
        call    Retardo         ;dejamos activo el tiempo necesario
        bcf      Servo2         ;bajamos flanco

        ;** SERVO3 **
        movlw   d'255'          ;Precargamos Posic con 255...
        movwf   Posic           ;...por si ha de funcionar en reverse mode
        movf    Posic3,W        ;Carga variable Posic con valor posición para servo
        btfsc   Modo,S3         ;Verifica si ha de funcionar en reverse mode
        goto    Inv3            ;Si. invertimos valor
        movwf   Posic           ;No
        goto    Sigue3
Inv3    subwf   Posic,F         ;Si. invertimos valor
Sigue3  movf    Offset3,w       ;Carga Offset del servo correspondiente
        movwf   Offset
        bsf      Servo3         ;activamos flanco
        call    Retardo         ;dejamos activo el tiempo necesario
        bcf      Servo3         ;bajamos flanco

        ;** SERVO4 **
        movlw   d'255'          ;Precargamos Posic con 255...
        movwf   Posic           ;...por si ha de funcionar en reverse mode
        movf    Posic4,W        ;Carga variable Posic con valor posición para servo
        btfsc   Modo,S4         ;Verifica si ha de funcionar en reverse mode
        goto    Inv4            ;Si. invertimos valor
        movwf   Posic           ;No
        goto    Sigue4
Inv4    subwf   Posic,F         ;Si. invertimos valor
```

```
Sigue4  movf    Offset4,w       ;Carga Offset del servo correspondiente
        movwf   Offset
        bsf      Servo4         ;activamos flanco
        call    Retardo         ;dejamos activo el tiempo necesario
        bcf      Servo4         ;bajamos flanco

        ;** SERVO5 **
        movlw   d'255'          ;Precargamos Posic con 255...
        movwf   Posic           ;...por si ha de funcionar en reverse mode
        movf    Posic5,W        ;Carga variable Posic con valor posición para servo
        btfsc   Modo,S5         ;Verifica si ha de funcionar en reverse mode
        goto    Inv5            ;Si. invertimos valor
        movwf   Posic           ;No
        goto    Sigue5
Inv5    subwf   Posic,F         ;Si. invertimos valor
Sigue5  movf    Offset5,w       ;Carga Offset del servo correspondiente
        movwf   Offset
        bsf      Servo5         ;activamos flanco
        call    Retardo         ;dejamos activo el tiempo necesario
        bcf      Servo5         ;bajamos flanco

        ;** SERVO6 **
        movlw   d'255'          ;Precargamos Posic con 255...
        movwf   Posic           ;...por si ha de funcionar en reverse mode
        movf    Posic6,W        ;Carga variable Posic con valor posición para servo
        btfsc   Modo,S6         ;Verifica si ha de funcionar en reverse mode
        goto    Inv6            ;Si. invertimos valor
        movwf   Posic           ;No
        goto    Sigue6
Inv6    subwf   Posic,F         ;Si. invertimos valor
Sigue6  movf    Offset6,w       ;Carga Offset del servo correspondiente
        movwf   Offset
        bsf      Servo6         ;activamos flanco
        call    Retardo         ;dejamos activo el tiempo necesario
        bcf      Servo6         ;bajamos flanco

        ;** SERVO7 **
        movlw   d'255'          ;Precargamos Posic con 255...
        movwf   Posic           ;...por si ha de funcionar en reverse mode
        movf    Posic7,W        ;Carga variable Posic con valor posición para servo
        btfsc   Modo,S7         ;Verifica si ha de funcionar en reverse mode
        goto    Inv7            ;Si. invertimos valor
        movwf   Posic           ;No
        goto    Sigue7
Inv7    subwf   Posic,F         ;Si. invertimos valor
Sigue7  movf    Offset7,w       ;Carga Offset del servo correspondiente
        movwf   Offset
        bsf      Servo7         ;activamos flanco
        call    Retardo         ;dejamos activo el tiempo necesario
        bcf      Servo7         ;bajamos flanco

Rest    bsf     SSPCON,CKP      ;Activamos reloj I2C
        banksel PORTA
        bsf     Led             ;apaga led de actividad I2C

        ;Restauramos las copias de los registros
        banksel BkStatus2
        movf    BkStatus2,W     ;Restaura las copias de registros
        movwf   STATUS          ;registro de estado
        movf    BkW2,W          ;registro W

        retfie



; -----------------------------------------------------------------------------------


INICIO          ;Inicio del cuerpo del programa


        banksel TRISA
        movlw   b'00101111'     ;A0,1,2,3,5 Entradas (Analógicas) , A4 Salida (Led)
        movwf   TRISA           ;
        movlw   b'00111100'     ;B0,B1 Salida (Servos), B2,B3,B4,B5 switches dip. B6,B7 prog
```

```
        movwf   TRISB           ;
        movlw   b'00000000'     ;C0,C1,C2,C5,C6,C7 Salida (Servos). C3,C4 I2C
        movwf   TRISC           ;


        ;Configuración para interrupciones por overflow de TMR0

        banksel OPTION_REG
        movlw   b'10000111'     ;Configuracion OPTION para TMR0 (Prescaler 1:256)
        movwf   OPTION_REG
        movlw   b'10100000'     ;Establece interrupciones
        movwf   INTCON          ;activas para overflow de TMR0
        bcf     INTCON,GIE      ;..pero no la general de momento
        banksel TMR0
        movlw   d'177'          ;Activa el TMR0 con 177 (complemento de 78)
         movwf  TMR0            ;256*78=19.968 (casi 20.000 usg= 20ms)

        ;Configuración para gestión i2c
        call    init_i2c_Slave ;Configuración para uso de i2c
        banksel INTCON
        bsf     INTCON,GIE      ;Activamos las interrupciones


        banksel Posic1

        ;Posiciones por defecto de los servos
        clrf    Posic0
        clrf    Posic1
        clrf    Posic2
        clrf    Posic3
        clrf    Posic4
        clrf    Posic5
        clrf    Posic6
        clrf    Posic7

        ;Valores por defecto de los Offsets
        movlw   d'70'
        movwf   Offset0
        movwf   Offset1
        movwf   Offset2
        movwf   Offset3
        movwf   Offset4
        movwf   Offset5
        movwf   Offset6
        movwf   Offset7

        ;En Principio todos los servos funcionaran sin reverse mode
        clrf    Modo

        clrf    StatI2C

        ;Todas las entradas analógicas están a 0
        clrf    PORTA
        clrf    APort
        clrf    Analog
        clrf    Analog0
        clrf    Analog1
        clrf    Analog2
        clrf    Analog3
        clrf    Analog4


BUCLE


        ;Leemos puerto analógico 0
        banksel APort
        clrf    APort
        call    ReadAnalog
        movf    Analog,W
        movwf   Analog0

        ;Leemos puerto analógico 1
        incf    APort,F
        call    ReadAnalog
        movf    Analog,W
```

```
        movwf   Analog1

        ;Leemos puerto analógico 2
        incf    APort,F
        call    ReadAnalog
        movf    Analog,W
        movwf   Analog2

        ;Leemos puerto analógico 3
        incf    APort,F
        call    ReadAnalog
        movf    Analog,W
        movwf   Analog3

        ;Leemos puerto analógico 4
        incf    APort,F
        call    ReadAnalog
        movf    Analog,W
        movwf   Analog4


goto BUCLE




;*****************************************************************************
; SUBRUTINAS
;*****************************************************************************


; -----------------------------------------------------------------------------------
Retardo         ;Provoca un retardo segun el valor de "Posic" y "Offset". Su valor en
                ;usg será: Retardo = 10 + 5xOffset + 8xPosic
; -----------------------------------------------------------------------------------

;Chequeo inicial: Delay fijo de 4usg (4 ciclos)
;-------------

        movf    Posic,F         ;Checkeamos si el valor es cero
        btfsc   STATUS,Z        ;
        goto    DelFijo         ;Si es cero salta a la parte de delay fijo
        NOP


;Primera parte: Delay variable en función Posic (entre 0 y 255). Ciclos=Posic x 8
;--------------

PLoop0  btfsc  PIR1,SSPIF       ; Ha ocurrido un evento SSP? (I2C)
        bcf    SSPCON,CKP       ; Si. Detenemos reloj I2C para evitar desbordamiento o timeout
        NOP
        NOP
        NOP
        decfsz  Posic,F         ; 1 + (1) es el tiempo 0  ?
         goto    PLoop0         ; 2 no, loop
        NOP


;Segunda parte: Delay fijo dependiente del valor de Offset. Ciclos=6+5xOffset
;-------------

DelFijo
        movf    Offset,W        ; 1 set numero de repeticion
         movwf  PDel0           ; 1 |
SLoop0  btfsc  PIR1,SSPIF       ; Ha ocurrido un evento SSP? (I2C)
        bcf    SSPCON,CKP       ; Si. Detenemos reloj I2C para evitar desbordamiento o timeout
         decfsz PDel0, 1        ; 1 + (1) es el tiempo 0  ?
         goto   SLoop0          ; 2 no, loop
         return                 ; 2+2 Fin.



;-----------------------------------------------------------------------------
init_i2c_Slave          ;Inicializa valores para uso de I2C en Slave
```

```
                        ;Ha de ser llamado tras definir TRISC (de ser necesario)
;-------------------------------------------------------------------------------

        ;Guardamos copia de algunos registros
        movwf   BkW             ;Hace copia de W
        movf    STATUS,W        ;Hace copia de registro de estado
        banksel BkStatus
        movwf   BkStatus

        ;Establecemos dirección del esclavo segun switches dip (B2 y B3)
        movlw   b'01110000'
        movwf   DirNodo
        btfsc   PORTB,2
        bsf     DirNodo,2
        btfsc   PORTB,3
        bsf     DirNodo,1

        ;Configuramos I2C
        banksel TRISC           ; Pasamos a direccionar Banco 1
        movlw   b'00011000'     ; Establece líneas SDA y SCL como entradas...
        iorwf   TRISC,F         ;..respetando los valores para otras líneas.
        bcf     SSPSTAT,CKE     ; Establece I2C input levels
        bcf     SSPSTAT,SMP     ; Habilita slew rate
        bsf     SSPCON2,GCEN    ; Habilita direccionamiento global
        banksel DirNodo
        movf    DirNodo,W       ; Dirección esclavo
        banksel SSPADD
        movwf   SSPADD          ;
        banksel SSPCON                  ; Pasamos a direccionar Banco 0
        movlw   b'00110110'     ; Slave mode, SSP enable,
        movwf   SSPCON          ;
        bcf     PIR1,SSPIF      ; Limpia flag de eventos SSP
        bcf     PIR1,7          ; Limpia bit. Mandatorio por Datasheet

        ;Configuración para interrupciones por evento I2C
        banksel PIE1
        bsf     PIE1,SSPIE
        bsf     INTCON,PEIE

        ;Restauramos las copias de los registros
        movf    BkStatus,W      ;Restaura las copias de registros
        movwf   STATUS          ;registro de estado
        movf    BkW,W           ;registro W

        return


; ----------------------------------------------------------------------------------------
SSP_Handler     ; Este manejador controla cada evento SSP (I2C) acontecido.
                ; El código que se muestra abajo chequea 5 posibles estados.
                ; Cada uno de los 5 estados SSP son identificados haciendo
                ; XOR de los bits del registro SSPSTAT con mascaras de bits
                ; predeterminadas. Una vez que el estado ha sido identificado
                ; se llevan a cabo las acciones pertinentes. Los estados
                ; indefinidos son considerados como estados de error.

                ; State 1: Operación de escritura I2C, ultimo byte era de dirección.
                ; SSPSTAT bits: S = 1, D_A = 0, R_W = 0, BF = 1

                ; State 2: Operación de escritura I2C, ultimo byte era de datos.
                ; SSPSTAT bits: S = 1, D_A = 1, R_W = 0, BF = 1

                ; State 3: Operación de lectura I2C, ultimo byte era de dirección.
                ; SSPSTAT bits: S = 1, D_A = 0, R_W = 1, BF = 0

                ; State 4: Operación de lectura I2C, ultimo byte era de datos.
                ; SSPSTAT bits: S = 1, D_A = 1, R_W = 1, BF = 0

                ; State 5: Reset lógico del Slave I2C por NACK del master.
                ; SSPSTAT bits: S = 1, D_A = 1, R_W = 0, BF = 0

; ----------------------------------------------------------------------------------------


        banksel SSPSTAT
        movf    SSPSTAT,W       ; Obtiene el valor de SSPSTAT
```

```
        andlw   b'00101101'     ; elimina los bits no importantes SSPSTAT.
        banksel Temp
        movwf   Temp            ; para chequeo posterior.

State1:                         ; Operación de escritura, ultimo byte ha sido
        movlw   b'00001001'     ; de dirección, el buffer está lleno.
        banksel Temp
        xorwf   Temp,W          ;
        btfss   STATUS,Z        ; Estamos en el primer estado?
        goto    State2          ; No, checkeamos siguiente estado
        call    ReadI2C         ; SI. Hacemos un read SSPBUF (para vaciar buffer).
        bcf     StatI2C,0       ; Limpiamos flags de control de ...
        bcf     StatI2C,1       ; ...llegada de tramas
                                ; El Hardware se ocupa de mandar Ack
        return

State2:                         ; Operación de escritura, ultimo byte ha sido
        movlw   b'00101001'     ; de datos, el buffer está lleno.
        banksel Temp
        xorwf   Temp,W
        btfss   STATUS,Z        ; Estamos en el segundo estado?
        goto    State3          ; NO, checkeamos siguiente estado
        call    ReadI2C         ; SI, Tomamos el byte del SSP.

        ;Aquí tenemos en W el valor del dato recibido
        movwf   MensajeIn

        btfsc   StatI2C,0       ;Chequeamos que dato es (registro, nºdispositivo o valor)
        goto    RegYaLlego      ;Registro ya llegó así que es nºservo o valor
        movf    MensajeIn,W     ;Es Registro
        movwf   Registro
        bsf     StatI2C,0
        return

RegYaLlego
        btfsc   StatI2C,1       ;Chequeamos que dato es (nºdispositivo o valor)
        goto    DisYaLlego      ;nºdispositivo ya llegó así que es valor
        movf    MensajeIn,W     ;Es nºdispositivo
        movwf   DispNo
        bsf     StatI2C,1
        return

DisYaLlego      ;Sabemos el registro, nºdispositivo y tenemos el dato, actualizamos dato del registro
        call    UpdateReg
        return

State3:                         ; Operación de lectura, ultimo byte ha sido
        movlw   b'00001100'     ; de dirección, el buffer está vacío
        banksel Temp
        xorwf   Temp,W
        btfss   STATUS,Z        ; Estamos en el tercer estado?
        goto    State4          ; NO, checkeamos siguiente estado
                                ; SI
        btfsc   StatI2C,0       ;Chequeamos si ya sabemos el registro a leer
        goto    RegYaLlegoR     ;Lo sabemos,
        movlw   0               ;No lo sabemos, devolvemos un cero por defecto
        call    WriteI2C        ;escribimos el byte en SSPBUF
        return
RegYaLlegoR
        btfsc   StatI2C,1       ;Chequeamos si ya sabemos el nºdispositivo a leer
        goto    DisYaLlegoR     ;Lo sabemos
        movlw   0               ;No lo sabemos, devolvemos un cero por defecto
        call    WriteI2C        ;escribimos el byte en SSPBUF
        return
DisYaLlegoR     ;Sabemos el registro y nºdispositivo a leer. Lo leemos y enviamos el dato
        call    ReadReg
        movf    MensajeOut,W
        call    WriteI2C        ; SI, escribimos el byte en SSPBUF
        bcf     StatI2C,0       ; Limpiamos flags de control de ...
        bcf     StatI2C,1       ; ...llegada de tramas
        return

State4:                         ; Operación de lectura, ultimo byte ha sido
        movlw   b'00101100'     ; de datos, el buffer está vacío
        banksel Temp
        xorwf   Temp,W
```

```
        btfss   STATUS,Z        ; Estamos en el cuarto estado?
        goto    State5          ; NO, checkeamos siguiente estado
                                ; SI. Operación no admitida.
        movlw   0               ; devolvemos un cero por defecto
        call    WriteI2C        ; escribimos el byte en SSPBUF
        return

State5:
        movlw   b'00101000'     ; Se ha recibido un NACK mientras se transmitían...
        banksel Temp
        xorwf   Temp,W          ; ..datos al master. La lógica del Slave..
        btfss   STATUS,Z        ; ..se resetea en este caso. R_W = 0, D_A = 1
        goto    I2CErr          ; y BF = 0
        return                  ; Si no estamos en State5, entonces es
                                ; que algo fue mal

I2CErr  nop                     ; Algo fue mal. Reseteamos el módulo I2C
        call    ReadI2C         ; Vaciamos buffer por si hubo overflow
        banksel SSPCON
        bcf     SSPCON,SSPEN    ; Detenemos I2C
        banksel SSPSTAT
        clrf    SSPSTAT
        bcf     SSPSTAT,CKE     ; Establece I2C input levels
        bcf     SSPSTAT,SMP     ; Habilita slew rate
        banksel SSPCON
        bsf     SSPCON,SSPEN    ; Reactivamos I2C

        return


;*****************************************************************************
WriteI2C        ;Usada por SSP_Handler para escribir datos en bus I2C
;*****************************************************************************

        banksel SSPCON
        movwf   SSPBUF          ; Escribe el dato en W
        bsf     SSPCON,CKP      ; Libera el reloj
        return

;*****************************************************************************
ReadI2C         ;Usada por SSP_Handler para escribir datos en bus I2C
;*****************************************************************************

        banksel SSPBUF
        movf    SSPBUF,W        ; Toma el byte y lo guarda en W
        return


;*****************************************************************************
UpdateReg               ;Actualiza Registro ordenado por I2C
;*****************************************************************************

        ;Procedemos a actuar según la orden recibida del Master. Haremos un Pseudo CASE
        ;que actualice solo los registros escribibles e ignore los que no se puedan escribir



M_01    ;xxxxxxxxx Modo xxxxxxxxx
        movlw   d'1'            ;
        xorwf   Registro,W      ;
        btfss   STATUS,Z        ; Es este el registro a actualizar?
        goto    M_02            ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si. procedemos a actualizar el registro
        movwf   Modo
        return                  ;Regresamos a la espera de una nueva orden

M_02    ;xxxxxxxxx Offset xxxxxxxxx
        movlw   d'2'            ;
        xorwf   Registro,W      ;
        btfss   STATUS,Z        ; Es este el registro a actualizar?
        goto    M_03            ; No, chequeamos siguiente caso
                                ; Si. procedemos a actualizar registro del servo correspondiente

O_00    movlw   d'0'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
```

```
        goto    O_01            ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Offset0
        return                  ;Regresamos a la espera de una nueva orden
O_01    movlw   d'1'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    O_02            ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Offset1
        return                  ;Regresamos a la espera de una nueva orden
O_02    movlw   d'2'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    O_03            ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Offset2
        return                  ;Regresamos a la espera de una nueva orden
O_03    movlw   d'3'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    O_04            ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Offset3
        return                  ;Regresamos a la espera de una nueva orden
O_04    movlw   d'4'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    O_05            ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Offset4
        return                  ;Regresamos a la espera de una nueva orden
O_05    movlw   d'5'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    O_06            ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Offset5
        return                  ;Regresamos a la espera de una nueva orden
O_06    movlw   d'6'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    O_07            ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Offset6
        return                  ;Regresamos a la espera de una nueva orden
O_07    movlw   d'7'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    O_Err           ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Offset7
        return                  ;Regresamos a la espera de una nueva orden
O_Err   return                  ;Error en numero de servo. No se hace nada


M_03    ;xxxxxxxxx Posicion xxxxxxxxx
        movlw   d'3'            ;
        xorwf   Registro,W      ;
        btfss   STATUS,Z        ; Es este el registro a actualizar?
        goto    M_Error         ; No, chequeamos siguiente caso
                                ; Si. procedemos a actualizar registro del servo correspondiente

S_00b   movlw   d'0'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    S_01b           ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Posic0
        return                  ;Regresamos a la espera de una nueva orden
S_01b   movlw   d'1'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    S_02b           ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Posic1
```

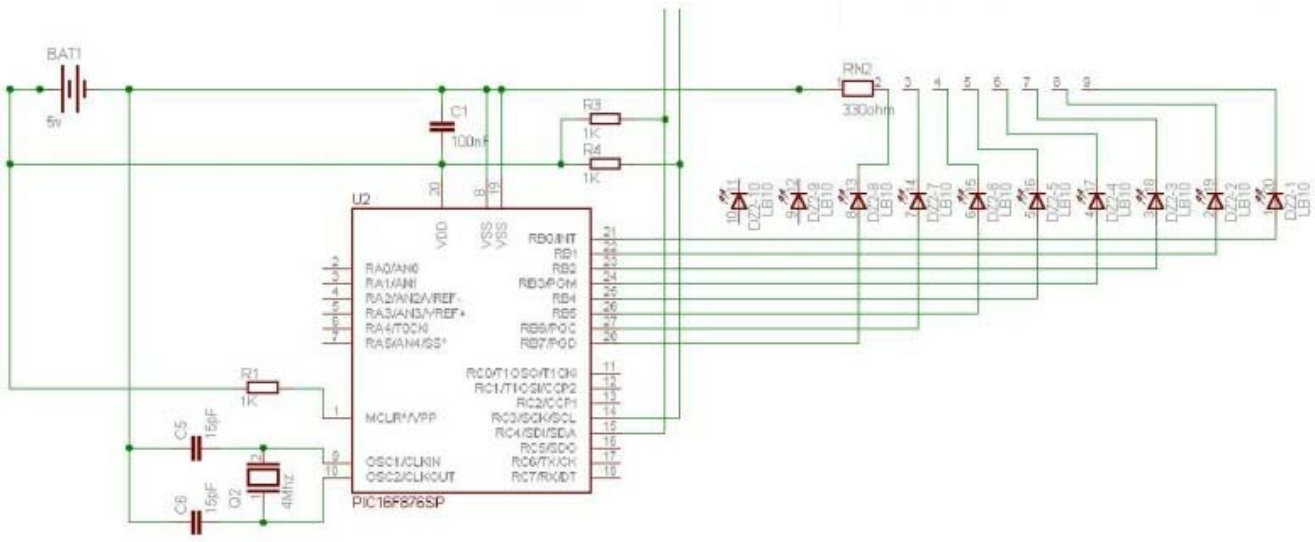
```
        return                  ;Regresamos a la espera de una nueva orden
S_02b   movlw   d'2'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    S_03b           ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Posic2
        return                  ;Regresamos a la espera de una nueva orden
S_03b   movlw   d'3'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    S_04b           ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Posic3
        return                  ;Regresamos a la espera de una nueva orden
S_04b   movlw   d'4'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    S_05b           ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Posic4
        return                  ;Regresamos a la espera de una nueva orden
S_05b   movlw   d'5'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    S_06b           ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Posic5
        return                  ;Regresamos a la espera de una nueva orden
S_06b   movlw   d'6'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    S_07b           ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Posic6
        return                  ;Regresamos a la espera de una nueva orden
S_07b   movlw   d'7'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el servo a actualizar?
        goto    S_Errb          ; No, chequeamos siguiente caso
        movf    MensajeIn,W     ; Si, actualizamos su registro
        movwf   Posic7
        return                  ;Regresamos a la espera de una nueva orden
S_Errb  return                  ;Error en numero de servo. No se hace nada


M_Error ;No es un registro conocido o es de solo lectura. Se ignora.
        Return



;-----------------------------------------------------------------------
ReadReg         ;Leemos Registro solicitado por I2C
;-----------------------------------------------------------------------

        ;Haremos un Pseudo CASE que lea el registro solicitado y lo deje en MensajeOut



M_04R   ;xxxxxxxxxx Entrada analógica xxxxxxxxxx
        movlw   d'4'            ;
        xorwf   Registro,W      ;
        btfss   STATUS,Z        ; Es este el registro a leer?
        goto    M_07R           ; No, chequeamos siguiente caso
                                ; Si. procedemos a leer el registro
A_00    movlw   d'0'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el puerto analógico a leer?
        goto    A_01            ; No, chequeamos siguiente caso
        movf    Analog0,W       ; Si. procedemos a leer el registro
        movwf   MensajeOut      ; y lo mandamos al master
        return                  ; Regresamos a la espera de una nueva orden
A_01    movlw   d'1'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el puerto analógico a leer?
```

```
        goto    A_02            ; No, chequeamos siguiente caso
        movf    Analog1,W       ; Si. procedemos a leer el registro
        movwf   MensajeOut      ; y lo mandamos al master
        return                  ; Regresamos a la espera de una nueva orden
A_02    movlw   d'2'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el puerto analógico a leer?
        goto    A_03            ; No, chequeamos siguiente caso
        movf    Analog2,W       ; Si. procedemos a leer el registro
        movwf   MensajeOut      ; y lo mandamos al master
        return                  ; Regresamos a la espera de una nueva orden
A_03    movlw   d'3'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el puerto analógico a leer?
        goto    A_04            ; No, chequeamos siguiente caso
        movf    Analog3,W       ; Si. procedemos a leer el registro
        movwf   MensajeOut      ; y lo mandamos al master
        return                  ; Regresamos a la espera de una nueva orden
A_04    movlw   d'4'
        xorwf   DispNo,W
        btfss   STATUS,Z        ; Es este el puerto analógico a leer?
        goto    A_Error         ; No, chequeamos siguiente caso
        movf    Analog4,W       ; Si. procedemos a leer el registro
        movwf   MensajeOut      ; y lo mandamos al master
        return                  ; Regresamos a la espera de una nueva orden

A_Error ;No es un puerto analógico válido. Se devuelve valor 0
        clrf    MensajeOut
        return

M_07R   ;xxxxxxxxxx Version fimware xxxxxxxxxx
        movlw   d'7'            ;
        xorwf   Registro,W      ;
        btfss   STATUS,Z        ; Es este el registro a leer?
        goto    M_ErrorR        ; No, chequeamos siguiente caso
        movlw   Version         ; Si. procedemos a leer el registro (en este caso una constante)
        movwf   MensajeOut
        return                  ; Regresamos a la espera de una nueva orden

M_ErrorR        ;No es un registro conocido. Se devuelve valor 0
        clrf    MensajeOut
        return



;-------------------------------------------------------------------------------------------
ReadAnalog      ;Leemos el puerto analógico "APort" y dejamos su valor en "Analog"
;-------------------------------------------------------------------------------------------

        ;Copiamos el numero de puerto a una variable temporal
        banksel APort
        movf    APort,W
        movwf   APortTmp

        ;Configuración para uso de conversor A/D

        banksel ADCON1
        movlw   b'00101111'     ;A0,1,2,3,5 Entradas (Analógicas) , A4 Salida (Led)
        movwf   TRISA           ;
        movlw   b'00000000'     ;Todas las entradas son analógicas. Justif ADRESH
        movwf   ADCON1          ;
        banksel ADCON0
        movlw   b'11000001'     ;osci interno, activación módulo conversor
        bcf     STATUS,C        ;Limpiamos acarreo
        rlf     APortTmp,F      ;desplaza a la izda los bits de APortTmp para que los..
        rlf     APortTmp,F      ;..3 bits indicadores del puerto estén en la misma posición..
        rlf     APortTmp,F      ;..que CHS0...CHS2 de ADCON0...
        iorwf   APortTmp,W      ;..Para incluirlos en la configuración de lectura
        movwf   ADCON0          ;
        movlw   d'2'            ;Pausa para que de tiempo al condensador interno...
        movwf   Pausa           ;..a capturar el valor analógico
        call    HacerTiempo

        banksel ADCON0
        bsf     ADCON0,GO       ;Hace medición de presión
```

```
AD_W    btfsc   ADCON0,GO_DONE ;Conversión finalizada?
        goto    AD_W           ;No. Seguimos esperando
        movf    ADRESH,W       ;Si. Tomamos valor
        movwf   Analog
        return



;-------------------------------------------------------------------------------------------
HacerTiempo    ;realiza una pausa del numero de centesimas de segundo especificadas en "Pausa"
               ;El tiempo real es aproximado, dependiendo del número de interrupciones
               ;que se producan.
;-------------------------------------------------------------------------------------------

        movf    Pausa,W        ;Coloca el valor de pausa en BDel2...
        movwf   BDel2          ;...para no alterar su contenido

;.....................................................
; Generado con PDEL ver SP  r 1.0  el 24/02/03 Hs 18:31:22
; Descripcion: Delay 10000 ciclos (1 centésima de segundo)
;.....................................................
BCiclo  movlw     .8        ; 1 set numero de repeticion  (B)
        movwf     BDel0     ; 1 |
BLoop1  movlw     .249      ; 1 set numero de repeticion  (A)
        movwf     BDel1     ; 1 |
BLoop2  nop                 ; 1 nop
        nop                 ; 1 ciclo delay
        decfsz    BDel1, 1  ; 1 + (1) es el tiempo 0  ? (A)
        goto      BLoop2    ; 2 no, loop
        decfsz    BDel0,  1 ; 1 + (1) es el tiempo 0  ? (B)
        goto      BLoop1    ; 2 no, loop
BDelL1  goto BDelL2         ; 2 ciclos delay
BDelL2  nop                 ; 1 ciclo delay
;.....................................................
        decfsz  BDel2,F       ;Repite tantas veces el ciclo de una decima de segundo...
        goto    BCiclo        ;..como se lo indique BDel2
        return              ; 2+2 Fin.


Fin
        END
```

Ejemplo de Master

El siguiente sería un ejemplo de montaje electrónico para el Master que controlaría dicho servomotor.



A continuación se incluye un ejemplo de código para el Master, que de forma cíclica lee la entrada analógica número 0 y muestra su valor digital en la barrera de leds. Asímismo mueve los servos 1 y 6 de un lado a otro, a posiciones concretas. El código para control del Slave ha de incluirse en el cuerpo del programa (donde el bucle MLoop) y consiste simplemente en utilizar las rutinas que se han documentado al principio del programa. Se cargan los datos deseados en las variables indicadas para cada rutina y se llama a la rutina. En caso de variar la dirección del slave, será necesario variar la macro **#define DirSvI2C b'01110000'**

```
; msvi-02
; Master para control del módulo SvI2C
; Por: Alejandro Alonso Puig
; Fecha: 11/10/2003
; Controlador: 16F876 4Mhz
; Función:
; Controla por I2C el módulo slave de control de servos y puertos analógicos SvI2C
;
; Las subrutinas definidas para el intercambio de datos con el módulo SvI2C
; son los siguientes:
;
;
; ESCRITURA DE REGISTROS
; ----------------------
;
; SetPosSvI2C         Ordena por i2c al módulo SvI2C la actualización del registro
;                     de posición del servo "ServoSvI2C" según el valor de
;                     "PosSvI2C".
;
; SetOffsetSvI2C      Ordena por i2c al módulo SvI2C la actualización del registro
;                     de Offset del servo "ServoSvI2C" según el valor de
;                     "OffsetSvI2C".
;
; SetModoSvI2C        Ordena por i2c al módulo SvI2C la actualización del registro
;                     Mode que establece el sentodo de giro de los servos
;                     según el valor de "ModoSvI2C".


; LECTURA DE REGISTROS
; ----------------------
;
; ReadVerSvI2C        Obtiene por i2c la versión de firmware del módulo SvI2C
;                     y la deja en la variable "VerSvI2C"
;
; ReadAnalogSvI2C     Obtiene por i2c el valor digital (8bits) de la señal analógica
;                     presente en el puerto "APortSvI2C" del módulo SvI2C y lo
;                     deja en "AnalogSvI2C"



        list            p=16F876
        include         "P16F876.INC"


;Definición de constantes

        #define ClockValue      d'9'    ;(100khz) valor para cálculo de vel. I2C que pasará a SSPADD
        #define DirSvI2C        b'01110000'    ;Dirección Módulo SvI2C (01110000)

        ;Códigos de registro del módulo SvI2C
        #define SvI2CModo       d'1'    ;Modo de funcionamiento de servo (normal=0 o reverse=1)
        #define SvI2COffset     d'2'    ;Offset
        #define SvI2CPos        d'3'    ;Posicion
        #define SvI2CAnalog     d'4'    ;Entrada analógica
        #define SvI2CVer        d'7'    ;Versión Firmware


;Definición de variables

        cblock  0x20
        MensajeIn       ;Contendrá el dato recibido por I2C del slave
        MensajeOut      ;Contendrá el dato a enviar por I2C al slave
        DirSlave        ;Dirección del Slave
```

```
        BkStatus        ;Backup del registro STATUS
        BkW             ;Backup W

        BDel0           ;Usada en retardos
        BDel1           ;Usada en retardos
        BDel2           ;Usada en retardos
        Pausa           ;Usada en para hacer pausas con subr "HacerTiempo"


        ;Dispositivos del módulo SvI2C
        ServoSvI2C      ;Servo cuyo registro se accederá por i2c
        APortSvI2C      ;Puerto analógico

        ;Registros del módulo SvI2C
        PosSvI2C        ;Posición que se desea que tenga la servo
        OffsetSvI2C     ;Offset que se desea que tenga el servo
        ModoSvI2C       ;Registro de modo (normal/reverse)
        VerSvI2C        ;Versión Firmware
        AnalogSvI2C     ;Valor puerto analógico


        endc            ;Fin de definiciones


        org     0
        goto    INICIO
        org     5



;*******************************************************************************



INICIO          ;Inicio del cuerpo del programa

        banksel TRISA           ;Apunta a banco 1
        movlw   b'00011111'     ;Entrada (switches). Solo 5 inferiores
        movwf   TRISA           ;
        movlw   b'00000000'     ;Salida (Leds)
        movwf   TRISB           ;
        banksel PORTB           ;Apunta a banco 0
        clrf    PORTB           ;Limpia puerto B
        clrf    PORTA           ;Limpia puerto A

        call    init_i2c_Master         ;Configuración para uso de i2c

        clrf    MensajeIn
        clrf    MensajeOut

        movlw   d'10'           ;Pausa de 10 centésimas de segundo para que en...
        movwf   Pausa           ;...el arranque de tiempo a los slaves a quedar...
        call    HacerTiempo     ;..configurados adecuadamente.


        movlw   d'220'          ;posición
        movwf   PosSvI2C


MLoop

        ;Proceso para resetear el slave en caso de bloqueo de algún tipo
        call Send_RStart
        call Send_Stop
        banksel PORTB

        ;lee versión firmware y la muestra por PORTB
;       call    ReadVerSvI2C
;       movf    VerSvI2C,W
;       movwf   PORTB


        ;Ordena funcionamiento servo 2 y 4 en sentido reverse. Resto normal
;       movlw   b'00010100'
;       movwf   ModoSvI2C
```

```
;       call    SetModoSvI2C

        ;lee valor analógico puerto 0 y la muestra por PORTB
        movlw   d'0'
        movwf   APortSvI2C
        call    ReadAnalogSvI2C
        movf    AnalogSvI2C,W
        movwf   PORTB


        ;Ordena movimiento a posición concreta
        movlw   d'1'            ;Servo
        movwf   ServoSvI2C
        movlw   d'10'           ;posición
        movwf   PosSvI2C
        call    SetPosSvI2C

        ;pausa de 10 centésimas de segundo
        movlw   d'10'
        movwf   Pausa
        call    HacerTiempo

        ;Ordena movimiento a posición concreta
        movlw   d'6'            ;Servo
        movwf   ServoSvI2C
        movlw   d'180'          ;posición
        movwf   PosSvI2C
        call    SetPosSvI2C

        ;pausa de 10 centésimas de segundo
        movlw   d'10'
        movwf   Pausa
        call    HacerTiempo


;-----Ciclo2-------


        ;Ordena movimiento a posición concreta
        movlw   d'1'            ;Servo
        movwf   ServoSvI2C
        movlw   d'200'          ;posición
        movwf   PosSvI2C
        call    SetPosSvI2C

        ;pausa de 10 centésimas de segundo
        movlw   d'10'
        movwf   Pausa
        call    HacerTiempo

        ;Ordena movimiento a posición concreta
        movlw   d'6'            ;Servo
        movwf   ServoSvI2C
        movlw   d'10'           ;posición
        movwf   PosSvI2C
        call    SetPosSvI2C

        ;pausa de 10 centésimas de segundo
        movlw   d'10'
        movwf   Pausa
        call    HacerTiempo

        goto    MLoop




;*****************************************************************************
; SUBRUTINAS
;*****************************************************************************
```

```
;-------------------------------------------------------------------------------
SetPosSvI2C           ;Ordena por i2c al módulo SvI2C la actualización del registro
                      ;de posición del servo "ServoSvI2C" según el valor de
                      ;"PosSvI2C".
;-------------------------------------------------------------------------------

       banksel DirSlave
       movlw   DirSvI2C
       movwf   DirSlave
       movlw   SvI2CPos       ;Código de registro a acceder
       movwf   MensajeOut
       call    Send_Start     ;Envía condición de inicio
       banksel DirSlave
       call    Enviar         ;Envía por I2C dirección de Slave y nº registro
       movf    ServoSvI2C,W   ;Se deja el nº de servo en W para que...
       call    Send_Byte      ;...Send_Byte lo envíe por i2c
       movf    PosSvI2C,W     ;Se deja el valor de posición en W para que...
       call    Send_Byte      ;...Send_Byte lo envíe por i2c
       call    Send_Stop      ;Envía condición de stop
       banksel DirSlave
       return



;-------------------------------------------------------------------------------
SetOffsetSvI2C        ;Ordena por i2c al módulo SvI2C la actualización del registro
                      ;de Offset del servo "ServoSvI2C" según el valor de
                      ;"OffsetSvI2C".
;-------------------------------------------------------------------------------

       banksel DirSlave
       movlw   DirSvI2C
       movwf   DirSlave
       movlw   SvI2COffset    ;Código de registro a acceder
       movwf   MensajeOut
       call    Send_Start     ;Envía condición de inicio
       banksel DirSlave
       call    Enviar         ;Envía por I2C dirección de Slave y nº registro
       movf    ServoSvI2C,W   ;Se deja el nº de servo en W para que...
       call    Send_Byte      ;...Send_Byte lo envíe por i2c
       movf    OffsetSvI2C,W  ;Se deja el valor de Offset en W para que...
       call    Send_Byte      ;...Send_Byte lo envíe por i2c
       call    Send_Stop      ;Envía condición de stop
       banksel DirSlave
       return



;-------------------------------------------------------------------------------
SetModoSvI2C          ;Ordena por i2c al módulo SvI2C la actualización del registro
                      ;Mode que establece el sentodo de giro de los servos
                      ;según el valor de "ModoSvI2C".
;-------------------------------------------------------------------------------

       banksel DirSlave
       movlw   DirSvI2C
       movwf   DirSlave
       movlw   SvI2CModo      ;Código de registro a acceder
       movwf   MensajeOut
       call    Send_Start     ;Envía condición de inicio
       banksel DirSlave
       call    Enviar         ;Envía por I2C dirección de Slave y nº registro
       movlw   d'0'           ;Se deja cualquier valor de nºDispositivo porque..
       call    Send_Byte      ;...este dato no es tomado en cuenta por Slave
       movf    ModoSvI2C,W    ;Se deja el valor de Offset en W para que...
       call    Send_Byte      ;...Send_Byte lo envíe por i2c
       call    Send_Stop      ;Envía condición de stop
       banksel DirSlave
       return


;-------------------------------------------------------------------------------
ReadVerSvI2C          ;Obtiene por i2c la versión de firmware del módulo SvI2C
                      ;y la deja en la variable "VerSvI2C"
;-------------------------------------------------------------------------------
```

```
        banksel DirSlave
        movlw   DirSvI2C
        movwf   DirSlave
        movlw   SvI2CVer        ;Código de registro a acceder
        movwf   MensajeOut
        call    Send_Start      ;Envía condición de inicio
        banksel DirSlave
        call    Enviar          ;Envía por I2C dirección de Slave y n° registro
        movlw   d'0'            ;Se deja cualquier valor de n°Dispositivo porque..
        call    Send_Byte       ;...este dato no es tomado en cuenta por Slave
        call    Send_RStart     ;Envía condición de reinicio
        banksel DirSlave
        call    Recibir         ;Toma dato del Slave...
        movf    MensajeIn,W     ;...y lo guarda en...
        movwf   VerSvI2C        ;...la variable de registro correspondiente
        call    Send_Stop       ;Envía condición de stop
        banksel DirSlave

        return



;--------------------------------------------------------------------------------
ReadAnalogSvI2C             ;Obtiene por i2c el valor digital (8bits) de la señal analógica
                        ;presente en el puerto "APortSvI2C" del módulo SvI2C y lo
                        ;deja en "AnalogSvI2C"
;--------------------------------------------------------------------------------

        banksel DirSlave
        movlw   DirSvI2C
        movwf   DirSlave
        movlw   SvI2CAnalog     ;Código de registro a acceder
        movwf   MensajeOut
        call    Send_Start      ;Envía condición de inicio
        banksel DirSlave
        call    Enviar          ;Envía por I2C dirección de Slave y n° registro
        movf    APortSvI2C,W    ;Se envía el número de puerto analógico..
        call    Send_Byte       ;...a leer del slave
        call    Send_RStart     ;Envía condición de reinicio
        banksel DirSlave
        call    Recibir         ;Toma dato del Slave...
        movf    MensajeIn,W     ;...y lo guarda en...
        movwf   AnalogSvI2C     ;...la variable de registro correspondiente
        call    Send_Stop       ;Envía condición de stop
        banksel DirSlave

        return



;--------------------------------------------------------------------------------
init_i2c_Master             ;Inicializa valores para uso de I2C en Master
                        ;Ha de ser llamado tras definir TRISC y un valor para
                        ;ClockValue. Para frecuencia SCL=Fosc/(4x(ClockValue+1))
;--------------------------------------------------------------------------------

        ;Guardamos copia de algunos registros
        movwf   BkW             ;Hace copia de W
        movf    STATUS,W        ;Hace copia de registro de estado
        banksel PORTA
        movwf   BkStatus

        ;Configuramos I2C
        banksel TRISC           ; Pasamos a direccionar Banco 1
        movlw   b'00011000'     ; Establece líneas SDA y SCL como entradas...
        iorwf   TRISC,f         ;..respetando los valores para otras líneas.
        movlw   ClockValue      ; Establece velocidad I2C segun...
        movwf   SSPADD          ; ...valor de ClockValue
        bcf     SSPSTAT,6       ; Establece I2C input levels
        bcf     SSPSTAT,7       ; Habilita slew rate
        banksel SSPCON                ; Pasamos a direccionar Banco 0
        movlw   b'00111000'     ; Master mode, SSP enable, velocidad segun...
        movwf   SSPCON          ; ... Fosc/(4x(SSPADD+1))
        bcf     PIR1,SSPIF      ; Limpia flag de eventos SSP
        bcf     PIR1,7          ; Limpia bit. Mandatorio por Datasheet
```

```
        ;Restauramos las copias de los registros
        movf    BkStatus,W      ;Restaura las copias de registros
        movwf   STATUS          ;registro de estado
        movf    BkW,W           ;registro W

        return


; ----------------------------------------------------------------------------------------
Enviar  ;Envía un mensaje (comando) almacenado en "MensajeOut" al Slave cuya dirección
        ;se ha de encontrarse en la variable "DirSlave"
;----------------------------------------------------------------------------------------


        ;Guardamos copia de algunos registros
        movwf   BkW             ;Hace copia de W
        movf    STATUS,W        ;Hace copia de registro de estado
        banksel PORTA
        movwf   BkStatus

StEnv
        banksel DirSlave
        movf    DirSlave,W      ;Dirección esclavo
        call    Send_Byte       ;Envía dirección y orden de escritura
        call    WrtAckTest      ;Verifica llegada ACK
        banksel SSPCON2
        bcf     SSPCON2,ACKSTAT       ;limpia flag ACK
        xorlw   1
        btfss   STATUS,Z        ;Chequea si llegó ACK
        goto    SigueEnv        ;Si. Seguimos con envío dato
        call    Send_Stop       ;No. Reintentamos envío
        call    Send_Start
        goto    StEnv

SigueEnv
        banksel MensajeOut
        movf    MensajeOut,W    ;Lo deja en W para que la subrutina Send_Byte lo envíe

        call    Send_Byte       ;envía por i2c

        ;Restauramos las copias de los registros
        movf    BkStatus,W      ;Restaura las copias de registros
        movwf   STATUS          ;registro de estado
        movf    BkW,W           ;registro W

        return


; ----------------------------------------------------------------------------------------
Recibir ;Solicita dato al Slave cuya dirección ha de encontrarse en la variable
        ;"DirSlave" y lo mete en "MensajeIn".
;----------------------------------------------------------------------------------------

        ;Guardamos copia de algunos registros
        movwf   BkW             ;Hace copia de W
        movf    STATUS,W        ;Hace copia de registro de estado
        banksel PORTA
        movwf   BkStatus

StRec
        banksel DirSlave
        movf    DirSlave,W      ;Dirección esclavo
        iorlw   b'00000001'     ;con orden de lectura
        call    Send_Byte       ;Envía dirección y orden de lectura
        call    WrtAckTest      ;Verifica llegada ACK
        banksel SSPCON2
        bcf     SSPCON2,ACKSTAT       ;limpia flag ACK
        xorlw   1
        btfsc   STATUS,Z        ;Chequea si llegó ACK
        goto    StRec           ;No. Reintentamos envío
                                ;Si. Leemos dato
        call    Rec_Byte        ;Recibe dato por i2c y lo mete en "MensajeIn"

        ;Restauramos las copias de los registros
        movf    BkStatus,W      ;Restaura las copias de registros
```

```
        movwf   STATUS          ;registro de estado
        movf    BkW,W           ;registro W

        return



; ------------------------------------------------------------------------------------
Send_Start      ;Envía condición de start
;------------------------------------------------------------------------------------

        banksel SSPCON2
        bsf     SSPCON2,SEN     ; Envía Start
        call    CheckIdle       ;Espera fin evento
        return


; ------------------------------------------------------------------------------------
Send_RStart     ;Envía condición de Repeated Start
;------------------------------------------------------------------------------------

        banksel SSPCON2
        bsf     SSPCON2,RSEN    ; Envía Repeated Start
        call    CheckIdle       ;Espera fin evento
        return


; ------------------------------------------------------------------------------------
Send_Ack        ;Envía Ack
;------------------------------------------------------------------------------------

        banksel SSPCON2
        bcf     SSPCON2,ACKDT   ; acknowledge bit state to send (ack)
        bsf     SSPCON2,ACKEN   ; Inicia secuencia de ack
        call    CheckIdle       ;Espera fin evento
        return


; ------------------------------------------------------------------------------------
Send_Nack       ;Envía Nack para finalizar recepción
;------------------------------------------------------------------------------------

        banksel SSPCON2
        bsf     SSPCON2,ACKDT   ; acknowledge bit state to send (not ack)
        bsf     SSPCON2,ACKEN   ; Inicia secuencia de nack
        call    CheckIdle       ;Espera fin evento
        return


; ------------------------------------------------------------------------------------
Send_Stop       ;Envía condición de stop
;------------------------------------------------------------------------------------

        banksel SSPCON2
        bsf     SSPCON2,PEN     ;Activa secuencia de stop
        call    CheckIdle       ;Espera fin evento
        return


; ------------------------------------------------------------------------------------
Send_Byte       ;Envía el contenido de W por i2c
;------------------------------------------------------------------------------------

        banksel SSPBUF                  ; Cambia a banco 0
        movwf   SSPBUF          ; inicia condicion de escritura
        call    CheckIdle       ;Espera fin evento
        return



; ------------------------------------------------------------------------------------
Rec_Byte        ;Recibe dato por i2c y lo mete en "MensajeIn"
;------------------------------------------------------------------------------------

        banksel SSPCON2         ; Cambia a banco 1
        bsf     SSPCON2,RCEN    ; genera receive condition
```

```
        btfsc   SSPCON2,RCEN   ; espera a que llegue el dato
        goto    $-1
        banksel SSPBUF         ; Cambia a banco 0
        movf    SSPBUF,w       ; Mueve el dato recibido ...
        movwf   MensajeIn      ; ...  a MensajeIn
        call    CheckIdle      ;Espera fin evento
        return


; ---------------------------------------------------------------------------------------
CheckIdle       ;Chequea que la operación anterior termino y se puede proceder con
                ;el siguiente evento SSP
;---------------------------------------------------------------------------------------

        banksel SSPSTAT              ; Cambia a banco 1
        btfsc   SSPSTAT, R_W   ; Transmisión en progreso?
        goto    $-1
        movf    SSPCON2,W
        andlw   0x1F           ; Chequeamos con mascara para ver si evento en progreso
        btfss   STATUS, Z
        goto    $-3            ; Sigue en progreso o bus ocupado. esperamos
        banksel PIR1           ; Cambia a banco 0
        bcf     PIR1,SSPIF     ; Limpiamos flag
        return


;---------------------------------------------------------------------------------------
WrtAckTest      ;Chequea ack tras envío de dirección o dato
                ;Devuelve en W 0 o 1 dependiendo de si llegó (0) o no (1) ACK
;---------------------------------------------------------------------------------------

        banksel SSPCON2        ; Cambia a banco 1
        btfss   SSPCON2,ACKSTAT ;Chequea llegada ACK desde slave
        retlw   0              ;llegó ACK
        retlw   1              ;no llegó ACK


;---------------------------------------------------------------------------------------
HacerTiempo     ;realiza una pausa del numero de centesimas de segundo especificadas en "Pausa"

;---------------------------------------------------------------------------------------

        movf    Pausa,W        ;Coloca el valor de pausa en BDel2...
        movwf   BDel2          ;...para no alterar su contenido

;.............................................................
; Generado con PDEL ver SP  r 1.0  el 24/02/03 Hs 18:31:22
; Descripcion: Delay 10000 ciclos (1 centésima de segundo)
;.............................................................
BCiclo  movlw     .8        ; 1 set numero de repeticion  (B)
        movwf     BDel0     ; 1 |
BLoop1  movlw     .249      ; 1 set numero de repeticion  (A)
        movwf     BDel1     ; 1 |
BLoop2  nop                 ; 1 nop
        nop                 ; 1 ciclo delay
        decfsz    BDel1, 1  ; 1 + (1) es el tiempo 0  ? (A)
        goto      BLoop2    ; 2 no, loop
        decfsz    BDel0,  1 ; 1 + (1) es el tiempo 0  ? (B)
        goto      BLoop1    ; 2 no, loop
BDelL1  goto BDelL2         ; 2 ciclos delay
BDelL2  nop                 ; 1 ciclo delay
;.............................................................
        decfsz  BDel2,F        ;Repite tantas veces el ciclo de una decima de segundo...
        goto    BCiclo         ;..como se lo indique ADel2
        return              ; 2+2 Fin.


END
```