Práctica 5
Programación en C de CCS de operaciones de E/S de Microcontroladores PIC

- Introducción: Objetivos
- Ejemplo de programación de E/S con el compilador C de CCS
- Gestión de los puertos de E/S en C (CCS)
  - Directamente a través de la RAM
  - Mediante directivas y funciones
  - Mediante punteros
- Tareas a realizar

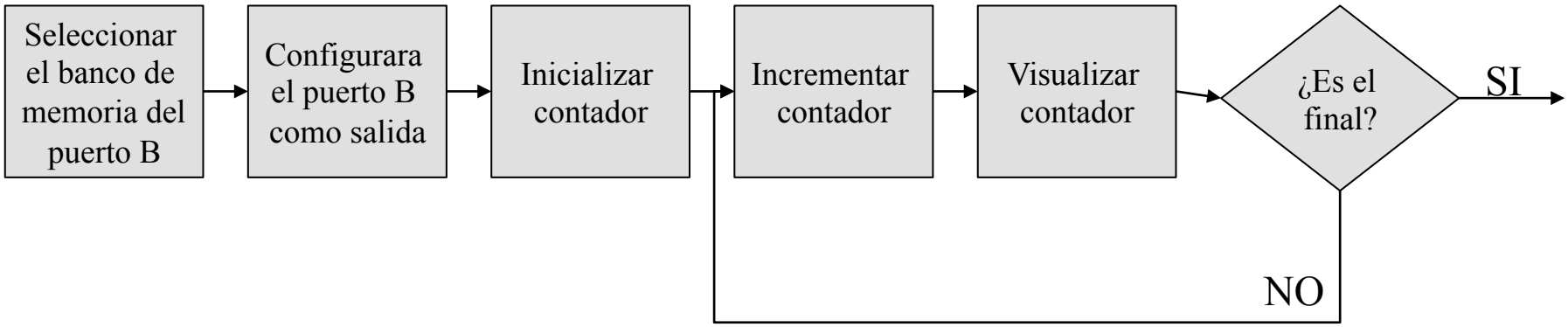
Introducción

Objetivos

- Conocer cómo se realiza la gestión de los puertos de E/S de los microcontroladores PIC mediante el lenguaje C (de CCS).
- Ser capaz de programar código en C de CCS que controle el PIC16F877A para realizar determinadas tareas relacionadas con los puertos de E/S.

Ejemplo (visto en práctica 2)

- Nos planteamos el siguiente problema:
  - Crearemos un programa para un PIC16F77A funcionando a 8MHZ encargado de contar hasta 0x5f. Cuando lo alcance se detendrá en un bucle no operativo. El valor del contador se visualizará en 8 diodos LED conectados al puerto B.



Ejemplo Práctica 2 en asm

```
#include <p16f877a.inc>
        LIST P=16f877a    ; Seleccionamos el micro
__config _WDT_OFF & _HS_OSC ; Deshabilitamos el WDT y fijamos el oscilador
; Asignación de etiquetas a registros.
f       EQU     0x01      ; registro f
portb   EQU     0x06      ; Dirección del registro del puerto B
estado  EQU     0x03      ; Dirección del registro de estado
conta   EQU     0x20      ; Lo usamos como variable contadora
        ORG     0         ; El programa comienza en la dirección 0
        GOTO    Inicio    ; salta a la dirección 5 para sobrepasar el vector INT.
        ORG     5
Inicio  BSF     estado,5 ; Selecciona banco 1 para llegar a TRISB
        MOVLW   0x00
        MOVWF   portb     ; Y se especifica que es de salida
        BCF     estado,5 ; Selección del banco 0 para trabajar con el puerto
        CLRF    conta     ; Ponemos nuestro contador a 0
bucle1  INCF    conta,f   ; conta + 1 --> conta (incrementa el contador)
        MOVF    conta,W   ; conta se carga en W
        MOVWF   portb     ; W se carga en el registro de datos del puerto B
        MOVLW   0x5f      ; W <-- 0x5f        (Final de cuenta deseado)
        SUBWF   conta,W   ; conta - W --> W. Si es cero, la cuenta está¡ acabada
        BTFSS   estado,2 ; Explora Z y si vale 1 es que W vale 0
                          ; se produce salto en ese caso por fin de cuenta
        GOTO bucle1       ; Si Z = 0 se vuelve a bucle1
bucle2  GOTO bucle2       ; Si Z = 1 se produce un bucle infinito
        END
```

Traducción Ejemplo Práctica 2 a C

```
#include <16f877a.h>                  //Seleccionamos el micro
#fuses HS,NOWDT                       //Establecemos configuración reloj, WDT
#use delay(clock= 8000000)            //Establecemos frecuencia
#byte portb=0x6                       //Definimos dirección puerto B
#byte trisb=0x86                      //Definimos dirección palabra conf. puerto B

void main()
{
  int conta=0;                        //Definimos e inicializamos contador
  trisb=0;                            //Configuramos puerto B como todo salidas
  while(conta<=0x5f)                  //Bucle hasta valor máximo contador
  {
   portb=conta;                       //Escribimos contador en puerto B
   delay_ms(500);                     //Retardo para apreciar cambio en B
   conta++;                           //Incrementamos contador
  }
  while(1);                           //Bucle infinito
}
```

Gestión de los puertos de E/S mediante C de CCS

- En el Tema 3 de la asignatura se describe la organización de los puertos de E/S de la MCU PIC16F877A.
- La gestión de dichos puertos mediante lenguaje ensamblador se abordó en la Práctica 2.
- A continuación se describirán métodos para gestionar dichos puertos mediante el lenguaje C de CCS.
- Concretamente, el lenguaje C de CCS permite varias formas de gestionar los puertos de E/S del la MCU, concretamente:
  - Directamente a través de la RAM
  - Mediante directivas y funciones específicas
  - Mediante punteros

Gestión mediante operación directa en la RAM

- Se basa en definir variables tipo byte (con la directiva **`#BYTE)`** en las direcciones de memoria de las palabras de control (TRISx) y de datos (PORTx) de los puertos. Ejemplos:

```
#BYTE TRISA=0x85
#BYTE PORTA=0x5
#BYTE TRISB=0x86
#BYTE PORTB=0x6
```

- Obviamente, es necesario un conocimiento exacto del mapa de memoria de la MCU concreta con la que se esté trabajando.

- Realmente, esto permite acceder directamente a otros registros clave de la MCU, como el de opciones o el de estado. Ejemplos:

```
#BYTE STATUS=0x03
#BYTE OPTION_REG=0x81
```

Gestión mediante operación directa en la RAM

- Los bits correspondientes de estos registros pueden activarse o testearse (esto es, escribirse o leerse) según se desee mediante escritura o lectura en las variables correspondientes, mediante variables tipo bit (directiva **`#BIT)`** o mediante las funciones para operar con bits. Ejemplos:

```
TRISB=0x00; //Puerto B configurado como todo salidas
PORTB=0x80; //Se activa el bit de mayor peso de B
-----------------------------------------------
#BIT TRISB7=TRISB.7
#BIT PORTB7=PORTB.7
TRISB7=FALSE; //Bit de mayor peso de B como salida
PORTB7=TRUE; //Se activa el bit de mayor peso de B
-----------------------------------------------
bit_clear(TRISB,7); //Bit de mayor peso de B como salida
bit_set(PORTB,7); //Se activa el bit de mayor peso de B
```

Gestión mediante operación directa en la RAM

- Ejemplo de código completo:

```
#include <16F877a.h>
#fuses HS,NOWDT
#use delay (clock =8000000 )        //Reloj de 8 MHz
#BYTE TRISB=0x86                    //TRISB en 86h
#BYTE PORTB=0x06                    //PORTB en 06h
#BYTE OPTION_REG=0x81               //OPTION_REG en 81h

void main() {

  bit_clear(OPTION_REG,7);          //Activa Pull-ups
  bit_set(TRISB,0);                 //RB0 entrada
  bit_clear(TRISB,1);               //RB1 salida
  bit_clear(PORTB,1);               //Inicia RB1 a cero

  while (TRUE)   {
   if (bit_test(PORTB,0)==1)        //RB0=1 -> RB1=0
         bit_clear(PORTB,1);
   else
         bit_set(PORTB,1);          //RB0=0 -> RB1=1
   }
}
```

Gestión mediante directivas y funciones específicas

- El compilador C de CCS proporciona funciones específicas para operar con los puertos de E/S:

  - **`OUTPUT_X (valor);`** . Escribe en el puerto X el valor indicado como parámetro.

  - **`INPUT_X();`** . Lee el puerto X y devuelve el valor leído como resultado de la función.

  - **`SET_TRIS_X(valor);`** . Escribe en la palabra de configuración del puerto X el valor indicado como parámetro.

  - **`GET_TRISX();`** . Lee el valor del registro TRISX y devuelve el valor leído como resultado de la función.

  - **`PORT_B_PULLUPS(booleano);`** . Activa (booleano a 1) o desactiva (booleano a 0) los pull-ups del puerto B.

Gestión mediante directivas y funciones específicas

- Además, existen funciones para operar directamente con los pines de la MCU:
  - **`OUTPUT_LOW (pin);`** . Pone a 0 el pin especificado.
  - **`OUTPUT_HIGH (pin);`** . Pone a 1 el pin especificado.
  - **`OUTPUT_BIT (pin,valor);`** . Pone a 0 ó 1 (según se indique en "**`valor`**") el pin especificado.
  - **`OUTPUT_TOGGLE(pin);`** . Invierte el valor de pin especificado.
  - **`OUTPUT_FLOAT (pin);`** . Pone el pin especificado en modo salida. Pone a uno salidas en drenador abierto.
  - **`INPUT_STATE(pin);`** . Lee el nivel de un pin sin cambiar su sentido.
  - **`INPUT(pin);`** . Lee el nivel de un pin.

Gestión mediante directivas y funciones específicas

- En las funciones anteriores, el valor de “pin” puede ser un número de pin o, equivalentemente, un identificador de pin. Estos identificadores habitualmente se definen en el fichero de cabecera correspondiente de la MCU. Por ejemplo, en el fichero 16F877a.h se definen (entre otros) los siguientes pines:

```
#DEFINE PIN_A0 40
#DEFINE PIN_B0 48
```

Por tanto, las siguientes expresiones serían equivalentes:

```
output_high(48);
output_high(PIN_B0);
```

Gestión mediante directivas y funciones específicas

- Sin embargo, el uso de las anteriores funciones específicas de E/S está condicionado por la declaración de las siguientes directivas del preprocesador:

  - **`#USE STANDARD_IO(port)`** . El compilador inserta automáticamente código para configurar como entrada o salida el pin del puerto **`port`** implicado en llamadas a función `output_high()` ó `input()`. Es el modo por defecto en el compilador CCS, que inserta código de configuración de sentido de E/S de cada función

  - **`#USE FAST_IO(port)`**. El usuario es el responsable de configurar el sentido de E/S en en puerto **`port`** mediante `set_tris_port()`.

  - **`#USE FIXED_IO(port_outputs=pin,...)`**. La propia directiva configura como salidas los pines del puerto **`port`** indicados en la lista (**`pin,..)`**.

Gestión mediante directivas y funciones específicas

- Ejemplo de código completo:

```
#include <16F877a.h>
#fuses HS,NOWDT
#use delay (clock =8000000 )        //Reloj de 8 MHz

#USE STANDARD_IO(B)                 //NO hace falta TRISB en el código

void main() {
  port_b_pullups (TRUE) ;           //Activa Pull-ups
  output_low(PIN_B1);               //Inicia RB1 a cero

  while (TRUE)    {
        if (input(PIN_B0)==1)       //RB0=1 -> RB1=0
        output_low(PIN_B1);
   else
        output_high(PIN_B1);        //RB0=0 -> RB1=1
   }
}
```

Gestión mediante directivas y funciones específicas

- Ejemplo de código completo:

```
#include <16F877a.h>
#fuses HS,NOWDT
#use delay (clock =8000000 )        //Reloj de 8 MHz

#USE FAST_IO(B)                     //SI hace falta TRISB en el código

void main() {
  port_b_pullups (TRUE) ;           //Activa Pull-ups
  output_low(PIN_B1);               //Inicia RB1 a cero
  set_tris_B(0x01);                 //Sentido del puerto B

  while (TRUE)    {
        if (input(PIN_B0)==1)       //RB0=1 -> RB1=0
        output_low(PIN_B1);
   else
        output_high(PIN_B1);        //RB0=0 -> RB1=1
   }
}
```

Gestión mediante directivas y funciones específicas

- Ejemplo de código completo:

```
#include <16F877a.h>
#fuses HS,NOWDT
#use delay (clock =8000000 )        //Reloj de 8 MHz

#USE FIXED_IO(B_outputs=PIN_B1)     //La directiva ejerce como TRISB

void main() {
  port_b_pullups (TRUE) ;           //Activa Pull-ups
  output_low(PIN_B1);               //Inicia RB1 a cero

  while (TRUE)    {
        if (input(PIN_B0)==1)       //RB0=1 -> RB1=0
        output_low(PIN_B1);
   else
        output_high(PIN_B1);        //RB0=0 -> RB1=1
   }
}
```

Gestión mediante punteros

- Se basa en definir punteros que apunten a las direcciones de memoria de las palabras de control (TRISx) y de datos (PORTx) de los puertos. Ejemplos:

```
#DEFINE TRISB (int*) 0x86
#DEFINE PORTB (int*) 0x6
```

- Como en el caso de gestión mediante operación directa con la RAM, es necesario un conocimiento exacto del mapa de memoria de la MCU concreta con la que se esté trabajando.

- Una vez definidos los punteros, se puede acceder a los registros y puertos mediante los operadores de C para punteros. Ejemplos:

```
valor=*PORTB //Se lee el puerto B y se guarda en valor
*PORTB=0xFF  //Se escribe en el puerto B
```

Tareas a realizar

Desarrollar las siguientes tareas, primero con simulación en C de CCS integrado en MPLAB y luego transfiriéndolo a la placa UNI DS3:

1. Realizar un programa para el PIC16F877A que lea el estado de los 6 interruptores E0-E5 conectados al puerto A y refleje el nivel lógico de los mismos sobre los leds S0-S5 conectados al puerto B.
2. Se pide realizar un programa para el PIC16F877A que active secuencialmente, de una en una, las ocho salidas de la puerta B (RB0-RB7), provocando un efecto de desplazamiento de derecha a izquierda.
3. Desarrollar una aplicación que encienda cada 0,5 segundos un led del puerto B (de menor a mayor peso), durante 0,5 segundos y posteriormente los leds correspondientes al puerto D (de mayor a menor peso). El proceso debe repetirse hasta que se active el pulsador correspondiente a RB0, quedando a partir de ese momento encendidos los leds del puerto B y D.
4. Desarrollar una aplicación que cuente el número de veces que se cierra el pulsador RB0 y las muestre en binario natural por el puerto D.

Deben entregarse los programas en C desarrollados para todas las tareas.