2.10 Mikroc PRO for PIC

Como ya hemos visto, hay varias divergencias entre los lenguajes mikroC y ANSI C. En este capítulo vamos a presentar las características específicas del mikroC con el propósito de facilitar la programación de los microcontroladores PIC.

ACCESO A LOS REGISTROS DE FUNCIONES ESPECIALES (SFR)

Como todos los microcontroladores, los de familia PIC tienen los registros de funciones especiales (SFR). Para programar un PIC, es necesario acceder a estos registros (para leerlos o escribir en ellos). A1 utilizar el compilador mikroC PRO for PIC es posible de acceder a cualquier SFR del microcontrolador de cualquier parte del código (los SFR se consideran como variables globales) sin necesidad de declararlo anteriormente. Los registros de funciones especiales se definen en un archivo externo e incluido dentro del compilador (archivo .def). Este archivo contiene todos los SFR del microcontrolador PIC a programar.

```
TRISB = 0; // todos los pines del puerto PORTB se configuran como salidas
PORTB = 0; // todos los pines del PORTB se ponen a 0
```

ACCESO A LOS BITS INDIVIDUALES

El compilador mikroC PRO for PIC le permite acceder a los bits individuales de variables de 8 bits por su nombre o su posición en byte:

```
INTCON.B0 = 0;  // Poner a 0 el bit 0 del registro INTCON
ADCON0.F5 = 1;  // Poner a 1 el bit 5 del registo ADCON0
INTCON.GIE = 0; // Poner a 0 el bit de interrupción global (GIE)
```

Para acceder a un bit individual, se puede utilizar '.FX' así como '.BX' (X es un entero entre 0 y 7 que representa la posición de bit).

TIPO SBIT

Si quiere declarar una variable que corresponde a un bit de un SFR, hay que utilizar el tipo sbit. Una variable de tipo sbit se comporta como un puntero y se debe declarar como una variable global:

```
sbit Botón_PARADA at PORTA.B7; // Botón_PARADA está definido
...
void main() {                  // Cualquier modificación de Botón_PARADA afectará a PORTA.B7
...                            // Cualquier modificación de PORTA.B7 afectará a Botón_PARADA
}
```

En este ejemplo, El Botón_PARADA es una variable declarada por el usuario, mientras que PORTA.B7 (bit 7 del puerto PORTA) será automáticamente reconocido por el compilador.

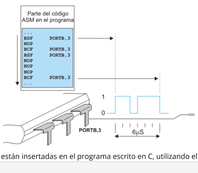
TIPO BIT

El compilador mikroC PRO for PIC proporciona un tipo de datos bit que se puede utilizar para declarar variables. No se puede utilizar en las listas de argumentos, punteros y los valores devueltos de funciones. Además, no es posible declarar e inicializar una variable de tipo bit en la misma línea. El compilador determina el bit en uno de los registros disponibles para almacenar las variables.

```
bit bf;    // Variable de tipo bit válida
bit *ptr;  // Variable de tipo bit inválida.
           // No hay punteros a una variable de tipo bit
bit bg = 0; // ERROR ¡ declaración con inicialización no está permitida
bit bg;
bg = 0;    // Declaración e inicialización válidas
```

INSERTAR CÓDIGO ASM EN C

A veces el proceso de escribir un programa en C requiere las partes del código escritas en ensamblador. Esto permite ejecutar las partes complicadas del programa de una forma definida con precisión en un período de tiempo exacto. Por ejemplo, cuando se necesita que los pulsos muy cortos (de unos microsegundos) aparezcan periódicamente en un pin del microcontrolador. En tales casos la solución más simple sería utilizar el código ensamblador en la parte del programa que controla la duración de pulsos.



Una o más instrucciones en ensamblador están insertadas en el programa escrito en C, utilizando el comando **asm**:

```
asm
{
  instrucciones en ensamblador
  ...
}
```

Los códigos escritos en ensamblador pueden utilizar constantes y variables anteriormente definidos en C. Por supuesto, como el programa entero está escrito en C, sus reglas se aplican al declarar estas constantes y variables.

```
unsigned char maximum = 100; // Declarar variables: maximum = 100
asm
{
  MOVF maximum,W             // Inicio del código ensamblador
                             // W = maximum = 100
  ...
}                            // Final del código ensamblador
```

FUNCIÓN DE INTERRUPCIÓN

Una interrupción detiene la ejecución normal de un programa para ejecutar las operaciones específicas. Una lista de sentencias a ejecutar debe estar escrita dentro de una función particular denominada interrupt(). La sintaxis de una interrupción en mikroC se parece a lo siguiente:

```
void interrupt() {
  cnt++ ;          // Al producirse una interrupción
                   // la cnt se incrementa en 1
  PIR1.TMR1IF = 0; // Poner a 0 el bit TMR1IF
}
```

A diferencia de las funciones estándar, no es necesario declarar el prototipo de la función interrupt(). Además, como la ejecución de esta función no forma parte de la ejecución de programa regular, no se debe llamar de ninguna parte de programa (se ejecutará automáticamente dependiendo de las condiciones que el usuario ha definido en el programa). En el siguiente capítulo vamos a dar una clara explicación de la ejecución y definición de subrutinas de interrupción.

LIBRERÍAS

Usted probablemente ha notado que en los ejemplos anteriores hemos utilizado algunas funciones como son 'Delay_ms', 'LCD_out', 'LCD_cmd' etc. Estas funciones están definidas en las librerías contenidas en el compilador mikroC. Una librería representa un código compilado, anteriormente escrito en mikroC, que contiene un conjunto de variables y funciones. Cada librería tiene un propósito específico. Por ejemplo, la librería LCD contiene funciones de visualización de la pantalla LCD, mientras queC_math proporciona algunas funciones matemáticas. Antes de utilizar alguna de ellas en el programa, es necesario comunicárselo al compilador al marcarlas en la lista de las librerías del compilador existentes. Si el compilador encuentra una función desconocida durante la ejecución de programa, primero va a buscar su declaración en las librerías marcadas.
Aparte de las librerías existentes, es posible crear las librerías y luego utilizarlas en el programa. El procedimiento de cómo crear librerías se describe en detalles en Help (Ayuda) del compilador. El compilador mikroC incluye tres tipos de librerías: - librerías ANSI C estándar:

LIBRARÍA	DESCRIPCIÓN
ANSI C Ctype Library	Utilizada principalmente para probar o para convertir los datos
ANSI C Math Library	Utilizada para las operaciones matemáticas de punto flotante
ANSI C Stdlib Library	Contiene las funciones de librerías estándar
ANSI C String Library	Utilizada para realizar las operaciones de cadenas y de manipulación de memoria

- librerías misceláneas:

LIBRARÍA	DESCRIPCIÓN
Button Library	Utilizada para desarrollar los proyectos
Conversion Library	Utilizada para la conversión de tipos de datos
Sprint Library	Utilizada para formatear los datos con facilidad
PrintOut Library	Utilizada para formatear los datos e imprimirlos
Time Library	Utilizada para cálculos de tiempo (formato UNIX time)
Trigonometry Library	Utilizada para la implementación de funciones trigonométricas fundamentales
Setjmp Library	Utilizada para los saltos de programa

- librerías para el hardware:

LIBRARÍA	DESCRIPCIÓN
ADC Library	Utilizada para el funcionamiento del convertidor A/D
CAN Library	Utilizada para las operaciones con el módulo CAN
CANSPI Library	Utilizada para las operaciones con el módulo CAN externo (MCP2515 o MCP2510)
Compact Flash Library	Utilizada para las operaciones con las tarjetas de memoria Compact Flash
EEPROM Library	Utilizada para las operaciones con la memoria EEPROM incorporada
EthernetPIC18Fxxj60 Library	Utilizada para las operaciones con el módulo Ethernet incorporado
Flash Memory Library	Utilizada para las operaciones con la memoria Flash incorporada
Graphic Lcd Library	Utilizada para las operaciones con el módulo LCD gráfico con resolución 128x64
I2C Library	Utilizada para las operaciones con el módulo de comunicación serial I2C incorporado
Keypad Library	Utilizada para las operaciones con el teclado (botones de presión 4x4)
Lcd Library	Utilizada para las operaciones con el LCD (de 2x16 caracteres)
Manchester Code Library	Utilizada para la comunicación utilizando el código Manchester
Multi Media Card Library	Utilizada para las operaciones con las tarjetas multimedia MMC flash
One Wire Library	Utilizada para las operaciones con los circuitos utilizando la comunicación serial One Wire
Port Expander Library	Utilizada para las operaciones con el extensor de puertos MCP23S17
PS/2 Library	Utilizada para las operaciones con el teclado estándar PS/2
PWM Library	Utilizada para las operaciones con el módulo PWM incorporado
RS-485 Library	Utilizada para las operaciones con los módulos utilizando la comunicación serial RS485
Software I2C Library	Utilizada para simular la comunicación I2C con software
Software SPI Library	Utilizada para simular la comunicación SPI con software
Software UART Library	Utilizada para simular la comunicación UART con software
Sound Library	Utilizada para generar las señales de audio
SPI Library	Utilizada para las operaciones con el módulo SPI incorporado
SPI Ethernet Library	Utilizada para la comunicación SPI con el módulo ETHERNET (ENC28J60)
SPI Graphic Lcd Library	Utilizada para la comunicación SPI de 4 bits con el LCD gráfico
SPI LCD Library	Utilizada para la comunicación SPI de 4 bits con el LCD (de 2x16 caracteres)
SPI Lcd8 Library	Utilizada para la comunicación SPI de 8 bits con el LCD
SPI T6963C Graphic Lcd Library	Utilizada para la comunicación SPI con el LCD gráfico
UART Library	Utilizada para las operaciones con el módulo UART incorporado
USB Hid Library	Utilizada para las operaciones con el módulo USB incorporado