# *Aprendiendo a programar Microcontroladores PIC en Lenguaje C con CCS*

www.edudevices.com.ar

**Por Andrés Raúl Bruno Saravia**

## Entrega Nº 5.

## ¿Cómo declaramos una variable en Lenguaje C?

En C siempre se deben declarar las variables. La declaración consiste en un tipo de dato, seguido por el nombre de la variable y el punto y coma:

```
int a;
int b,c,d;
int a = 10;
```

Los tres casos son definiciones correctas de variables, en el último además de declarar la variable se le asigna un valor inicial.
En caso de existir una expresión con variables de diferentes tipos, el resultado obtenido es del tipo de operando de mayor precisión.
**Todos los char se convierten a int.**
**Todos los float se convierten a double**.(hay que tener en cuenta que el tipo char es en realidad un int de menor precisión).

## ¿Cómo convertimos un tipo de una variable en otra?

A veces es útil, o necesario, realizar conversiones explícitas para obligar que una expresión sea de un cierto tipo.
La forma general de esta conversión en C es:

```
(tipo) expresión;
```

siendo **tipo**, el tipo de datos al que se convertirá la expresión.

**NOTA:** Esta conversión de tipos, (también llamada CAST), permite convertir expresiones no variables, esto es, si tengo una variable x de tipo int y le aplico (float)x lo que se convierte es el resultado, en caso que yo asigne esta operación a otra variable, pero no se convierte la variable x a float.
Supongamos que hacemos un programa que divide 10 por 3, uno sabe que el resultado será flotante: 3.333, y como 10 y 3 son enteros uno escribiría:

```
int a = 10, b = 3;
float r;
r = a/b;
printf("El resultado es %f", r);
```

Pero se encontraría que el resultado no es el deseado, esto ocurre porque en C la división entre enteros da como resultado un entero y en la realidad no siempre es así, (sólo en el caso que b sea divisor de a). Pero cambiando el cálculo de la división por:

```
R = (float)a/b;
```

Así se garantiza que el resultado será flotante.

## ¿Cómo definimos una constante numérica?

Las constantes se definen en diferentes formatos dependiendo del prefijo:

**0x (hexadecimal), 0 (octal), 0b (binario).**

## ¿Cómo definimos una constante de caracteres (string)?

Las variables almacenadas en memoria de programa suelen ser cadenas de caracteres. Este tipo de variable se define como: ***const rom char[]***

También se pueden declarar tablas de caracteres.

Ejemplo:

```
const rom char tabla[][20] = { "string 1", "string 2",
"string 3", "string 4" };
```

Tabla es una variable que contiene 80 caracteres en memoria de programa, ya que está compuesta de 4 cadenas de 20 caracteres cada una.

Para copiar una cadena de caracteres que se encuentra situada en la memoria de programa (ROM) en la memoria de datos (RAM), debemos usar una rutina o función estándar de C denominada **str2ram**

Veamos un ejemplo de copia de una cadena en RAM a otra en ROM:

```
void str2ram(static char *dest, static char rom *src)
{

while ((*dest++ = *src++) != '\0');
```

## ¿Que es la función printf()?

Es una función que está incluida en el archivo de cabecera **stdio.h** y se utilizan para enviar información por la **USART del microcontrolador.**
El número de parámetros pasados puede variar, dependiendo de la cantidad de variables a mostrar. El primer parámetro indica, por un lado, los caracteres que se mostrarán por pantalla y además los formatos que definen como se verán los argumentos, el resto de los parámetros son las variables a enviar por la USART.

**Ejemplo:**

```
int a = 100, b = 50;
printf("%i es mayor que %i\r", a, b);
```

Si conectamos a la USART de nuestro PIC un MAX232 y lo enlazamos al puerto COM de la PC, podremos ver la información con el programa Hypeterminal de Windows, que en este caso nos mostrará en la pantalla de la PC:

```
100 es mayor que 50
```

Los formatos más utilizados con printf() son:

| CODIGO | FORMATO |
|---|---|
| %c | Un solo carácter |
| %d | Decimal (un entero) |
| %i | Un entero |
| %f | Punto decimal flotante |
| %e | Notación científica |
| %o | Octal |
| %x | Hexadecimal |
| %u | Entero sin signo |
| %s | Cadena de caracteres |
| %% | Imprime un signo % |
| %p | Dirección de un puntero |

Los formatos pueden tener modificadores para especificar el ancho del campo, el número de lugares decimales y el indicador de alineación a la izquierda.

Ejemplos:

**%05d**, un entero de 5 dígitos de ancho; rellenará con ceros. Alineado a la derecha.
**%10.4f,** un real de 10 dígitos de ancho, con 4 decimales. Alineado a la derecha.
**%-10.2f**, un real de 10 dígitos de ancho, con 2 decimales. Alineado a la izquierda.

En la función **printf()** también se pueden encontrar **"caracteres de escape"** que permiten intercalar algún carácter especial en la cadena.

Ejemplo

```
printf("\nHola mundo.\n");
```

Aquí antes de imprimir el texto "Hola mundo", **n** obliga a un salto de línea - retorno de carro, (ENTER) dentro del Programa Hyperterminal, ya que estos caracteres especiales solo pueden sen entendidos y "vistos" por los programas emuladores de terminales.

Caracteres de escape mas usados:

| **CÓDIGO** | **DESCRIPCIÓN** |
|---|---|
| \n | Salto de línea – retorno de carro (ENTER) |
| \t | Tabulado horizontal |
| \v | Tabulado vertical |
| \r | Retorno de carro. |
| \b | Backspace. |
| \f | Alimentación de página. |

**Para usar la función printf() debemos previamente en la cabecera haber colocado la directiva #use RS232 ( parámetros), la cual debe encontrarse siempre después de la directiva #use delay. Esta directiva se usa para indicarle al compilador datos esenciales como ser: el BAUD RATE, los pines de la comunicación, el número de bits de la comunicación, si se usa la USART del PIC o se emula por software, etc. Un ejemplo de la directiva para un PIC16F887 que usa su USART sería la siguiente:**

```
#use RS232(BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6,
RCV=PIN_C7)
```

Un ejemplo de código completo es el siguiente programa en el cual transmitimos un **"hola mundo"** por la USART, y dicho mensaje podrá ser recibido por el programa Hypeterminal de Windows:

```
#include <16f887.h>
#fuses INTRC_IO,NOPROTECT,NOPUT,NOBROWNOUT,NOLVP
#use delay(clock=4000000)
#use RS232(BAUD=9600, BITS=8, PARITY=N, XMIT=PIN_C6,
RCV=PIN_C7)

void main(void)
{
   setup_adc_ports(NO_ANALOGS); //Selecciona terminales
   while(1)// ciclo for infinito
   {
     printf("Hola Mundo\r",temperatura);//imprimo x USART
     delay_ms(100);// delay
   }
}
```

***Continuará .....***