# 9.2 Naming Conventions

El propósito del presente documento es presentar el estándar que se debe de utilizar para codificar un programa en lenguaje C, independientemente de la plataforma que se trate se deben de seguir ciertas reglas al momento de desarrollar software, el objetivo de estas es que el programa sea claro y explícito.

**Convención para los nombres de variables**

Cada palabra de una variable debe empezar con Mayúscula después del prefijo del tipo de variable.

El uso del guión bajo podrá ser empleado para separar las palabras, por ejemplo:

u16Frec\_PWM

## Prefijo para el identificador de alcance de una variable.

El identificador o nombre de la variable deberá tener un prefijo que determine su alcance. El tipo de alcance podrá ser global o local. El alcance por default de una variable es **Global**. Este identificador deberá ser el primer carácter en cualquier nombre que se le vaya asignar a una variable.

|  |  |  |
| --- | --- | --- |
| **Identificador de alcance** | **Prefijo** | **Ejemplo** |
| Alcance local | L minúscula | uint8\_t u8Mot\_status |
|  | Ninguno | uint16\_t u16RPM |

## Prefijos para el nombre de una variable.

Los identificadores (nombre) de las variables deberán ser predefinidos con caracteres que representen el tipo de variable. Esto es representado en la siguiente tabla. A diferencia de la definición del tipo de variable que es con mayúscula, el nombre de la variable empieza con el tipo de variable en minúscula. Esto es:

int16\_t tipo de variable en mayúsculas.

i16Ctrl\_Out Nombre de la variable, empieza con el tipo pero en minúsculas.

| Caracteres empleados | Tipo Nombre Variable | Comentario |
| --- | --- | --- |
| u8 | uint8\_t u8Pulses; | unsigned char |
| i8 | int8\_t i8Input\_Value; | signed char |
| u16 | uint16\_t u16Frec\_Sensor; | unsigned short |
| i16 | int16\_t i16Integr\_Error; | signed short |
| u32 | uint32\_t u32Counts; | unsigned long |
| i32 | int32\_t i32Control\_Out; | signed long |
| f32 | float f32Value\_Filtered; | Float |
| f64 | double f64Variable; | Double |
| Array…………..a | uint16\_t au16Send\_Data[2] | uint16\_t es usado para determinar la longitud del arreglo. |
| Struct………….s | struct ssp\_err\_t sError; | Define sError como una estructura del tipo ssp\_err\_t |
| Register………Regn | #define Reg8TxD\_Data u0tb | Define el registro datos para Tx de n=8 bits |
| Pointer……………p | uint8\_t \* pu8Window; | Generalmente, p es combinado con otro carácter de prefijo. (En este ejemplo ‘pu8’ denota un puntero a un tipo uint8\_t.) |

## Convención para los nombres de las funciones.

El identificador de la función deberá ser **FN**. Después del prefijo FN debe de seguir un guión bajo, y posteriormente el tipo de dato que regresa la función.

El nombre de una función deberá consistir de un pronombre que describa el sujeto de la función y un verbo que describa la acción que realizará la función. Por ejemplo para la función que verifica si el dispositivo I2C está listo deberá ser nombrada.

uint16\_t FN\_u16PI\_Control(int16\_t li16Error)

## Convención para los nombres de las Subrutinas.

De forma similar a las funciones, se deberán de seguir las siguientes reglas para la definición de subrutinas.

El identificador de las subrutinas deberá ser **SR**.

Después del prefijo SR debe de seguir un guión bajo. Evitar separar el nombre de la subrutina con el guion bajo. Ejemplo:

void SR\_Motor\_Control(void)

A diferencia de las funciones una subrutina no regresa valor alguno, pero si puede contener parámetros en el llamado de esta como en una función.

## Convención general para nombres

Se deberá tener en cuenta al momento de nombrar ya sean variables o funciones, para que el nombre tenga la mayor claridad posible. Siguiendo los siguientes puntos:

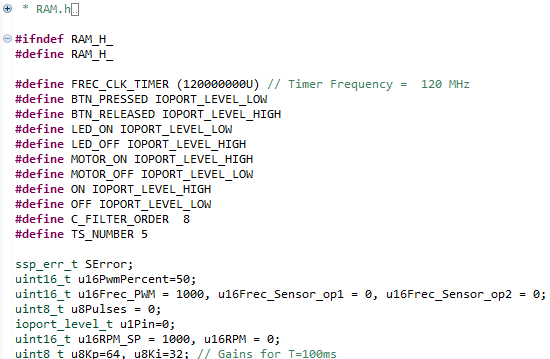
* Los nombres deberán ser lo suficientemente explícitos sin sobrecargar el uso de verbos. Por ejemplo, u8MessageBuffer es más significativo que u8Buf.
* El ahorrarse algunos caracteres y usar abreviaturas se debe de evitar, esto se reflejará al momento de leer y entender el código, lo cual tiene mayor prioridad que el ahorro de caracteres en el nombramiento de variables y funciones.

## 

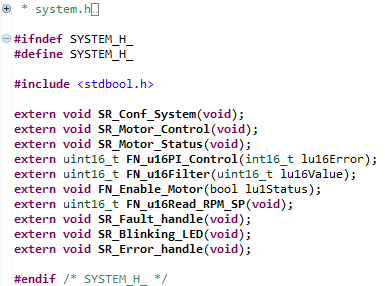
## Estructuración del programa

En esta sección se presenta la estructura general que debe de seguir cualquier programa implementado en lenguaje C en un microcontrolador.

Se incluyen los .h específicos del micro, las constantes y variables Globales en el encabezado RAM.h y además este RAM.h se usara en todos los .c



En los .h de cada .c se ponen como extern sus funciones y subrutinas para que los demás .c puedan hacer referencias a ellas y para que el propio .c las tenga como prototipo.



En cada .c se incluyen solo las referencias de los .h que contengan las funciones y subrutinas que necesitan

