

Tipos de datos				
Tipo	Denominación	Tamaño	Rango	Ejemplos de uso
uint8_t	Entero sin signo	8bits	[0,255]	Caracteres, contadores de for
uint16_t	Entero sin signo	16bits	[0,63535]	
uint32_t	Entero sin signo	32bits	[0,4294967295]	Almacenar valores de registros
int8_t	Entero con signo	8bits	[-128,127]	
int16_t	Entero con signo	16bits	[-32768,32767]	
int32_t	Entero con signo	32bits	[-2147483648,2147483647]	
float	Punto flotante	32bits	$[-3.4 \cdot 10^{38}, 3.4 \cdot 10^{38}]$	Hasta 6 dígitos significativos
double	Doble	64bits	$[-1.79 \cdot 10^{308}, 1.79 \cdot 10^{308}]$	Hasta 14 dígitos significativos

Sentencia if

```
if (condición){
    instrucción1;
    instrucción2;
}else if (condición2){
    instrucción3;
    instrucción4;
}else{
    instrucción5;
}
```

Las condiciones del *if* y del *else if* tienen que ser mutuamente excluyentes, es decir, no pueden darse simultáneamente.

El *else if* y el *else* no son obligatorios.

La condición **siempre** debe ir entre paréntesis.

Sentencia switch

```
switch (selector){
    case etiqueta1:
        sentencia1;
        break;
    case etiqueta2:
        sentencia2;
        break;
    case etiqueta3:
        sentencia3;
        break;
    default:
        sentencia;
}
```

El *selector* solo puede ser un entero o carácter.

Se deben recoger todas las opciones posibles en los *case* aunque sean iguales.

Es altamente recomendable utilizar el caso *default* para los casos que no estén definidos.

¡No olvidar el *break* para evitar seguir ejecutando instrucciones!

Bucle for

```
for (inicial; cond; increm){
    instrucciones;
}
```

PARA CADA incremento *DESDE* inicialización *MIENTRAS QUE* la condición sea cierta, *REALIZAR* instrucciones

Bucle while

```
while (condición){
    instrucciones;
}
```

MIENTRAS QUE la condición sea cierta, *REALIZAR* instrucciones

Bucle do while

```
do{
    instrucciones;
}while (condición);
```

REALIZAR las instrucciones *HASTA QUE* la condición sea *FALSA*
Se ejecuta al menos 1 vez



By **alejandrolara**

Published 4th March, 2024.

Last updated 4th June, 2024.

Page 2 of 5.

Sponsored by **Readable.com**

Measure your website readability!

<https://readable.com>

adc_lpc40xx.h

```
#define SEL_CANAL_0 (1u << 0)
#define SEL_CANAL_1 (1u << 1)
#define SEL_CANAL_2 (1u << 2)
#define SEL_CANAL_3 (1u << 3)
#define SEL_CANAL_4 (1u << 4)
#define SEL_CANAL_5 (1u << 5)
#define SEL_CANAL_6 (1u << 6)
#define SEL_CANAL_7 (1u << 7)
void adc_inicializar(uint32_t frecuencia,
uint32_t canales);
uint32_t adc_convertir(uint32_t canal);
float32_t adc_tension(uint32_t
resultado_adc);
```

glcd.h

```
#define NEGRO_RGB(0, 0, 0)
#define AZUL_OSCURO_RGB(0, 0, 128)
#define VERDE_OSCURO_RGB(0, 128, 0)
#define CIAN_OSCURO_RGB(0, 128, 128)
#define MARRON_RGB(128, 0, 0)
#define PURPURA_RGB(128, 0, 128)
#define OLIVA_RGB(128, 255, 0)
#define NARANJA_RGB(255, 128, 0)
#define GRIS_CLARO_RGB(192, 192, 192)
#define GRIS_OSCURO_RGB(128, 128, 128)
#define AZUL_RGB(0, 0, 255)
#define VERDE_RGB(0, 255, 0)
#define CIAN_RGB(0, 255, 255)
#define ROJO_RGB(255, 0, 0)
#define MAGENTA_RGB(255, 0, 255)
#define AMARILLO_RGB(255, 255, 0)
#define BLANCO_RGB(255, 255, 255)
void glcd_inicializar(void);
void glcd_borrar(uint16_t color);
int32_t glcd_xprintf(int32_t x,
int32_t y,
uint16_t color,
uint16_t color_fondo,
```

glcd.h (cont)

```
> uint32_t font,
const char *fmt, ...);
void glcd_circunferencia(int32_t xc,
int32_t yc,
int32_t radio,
uint16_t color);
void glcd_circulo(int32_t xc,
int32_t yc,
int32_t radio,
uint16_t color);
void glcd_punto(int32_t x,
int32_t y,
uint16_t color);
void glcd_linea(int32_t x0,
int32_t y0,
int32_t x1,
int32_t y1,
uint16_t color);
void glcd_rectangulo(int32_t x0,
int32_t y0,
int32_t x1,
int32_t y1,
uint16_t color);
void glcd_rectangulo_relleno(int32_t x0,
int32_t y0,
int32_t x1,
int32_t y1,
uint16_t color);
```

Máscaras binarias

Cambiar bit de dato		
1 en bit p	$(1 << p)$	$\text{dato} = (1 << p)$
0 en bit p	$\sim(1 << p)$	$\text{dato} \&= \sim(1 << p)$
Selec. bit p	$(\text{dato} >> p) \& 1$	



By **alejandrolara**

Published 4th March, 2024.
Last updated 4th June, 2024.
Page 3 of 5.

Sponsored by **Readable.com**
Measure your website readability!
<https://readable.com>

Operadores binarios (bit a bit)

	or
&	and
~	not
^	xor

Son operaciones bit a bit, no confundir con las operaciones booleanas que tienen como resultado TRUE o FALSE

Operadores booleanos

	or
&&	and
!	not
==	igual que
!=	diferente de

Se utilizan principalmente para definir condiciones en los if o bucles

Comentarios

//	Comentario de una línea
/* */	Comentario de bloque

Tipos de datos no estándar

unsigned char	uint8_t
unsigned short	uint16_t
unsigned int	uint32_t
unsigned long	uint64_t
char	int8_t
short	int16_t
int	int32_t
long	int64_t

Utilizar siempre los tipos estándar pues los aquí recogidos pueden variar entre versiones del compilador y el microcontrolador utilizado

Estructura general

```
#include <LPC407x_8x_177x_8x.h>
// Agregamos todos los #include que utilizemos
#include "lib rer ia.h "
// Agregamos todos los #define que sean necesarios
// porque sean constantes como PI o parámetros de
```

Estructura general (cont)

```
> // configuración
#define etiqueta valor
//Prototipos de funciones
int main(void){
    // Declaramos las variables que necesitamos
    uint8_t c = 0;
    // Inicializamos los periféricos y
    // componentes externos:
    glcd_inicializar();
    // Secuencia de instrucciones previas al
    // bucle infinito
    while(1){
        // Instrucciones a repetir infinitamente
    }
}
```

iocon_lpc40xx.h

```
void iocon_configurar_pin(LPC_GPIO_TypeDef
*gpio_regs,
    uin t32_t mascar a_pin,
    fun cio n_pin_t funcion,
    uin t32_t config ura cio n_es);
```

timer_lpc40xx.h

```
void timer_inicializar(LPC_TIM_TypeDef
*timer_regs);
void timer_retard o_m s(L PC_ TIM_ Ty peDef
*timer _regs,
    uin -
t32_t retard o_e n_ms);
void timer_retard o_u s(L PC_ TIM_ Ty peDef
*timer _regs,
    uin -
t32_t retard o_e n_us);
void timer_ini cia r_c icl os_ ms( LPC _TI M_T -
ypeDef *timer _regs,
    uin t32_t period o_e n_ms);
void timer_ini cia r_c icl os_ us( LPC _TI M_T -
ypeDef *timer _regs,
    uin t32_t period o_e n_us);
void timer_esp era r_f in_ cic lo( LPC _TI M_T -
ypeDef *timer _regs);
void timer_ini cia r_c ont eo_ ms( LPC _TI M_T -
ypeDef *timer _regs);
void timer_ini cia r_c ont eo_ us( LPC _TI M_T -
ypeDef *timer _regs);
uint32_t timer_lee r(L PC_ TIM_ Ty peDef *timer -
_regs);
```

