Informática Industrial GIEI UCA Cheat Sheet

by alejandrolara via cheatography.com/201385/cs/42628/

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	$[-3.4\cdot10^{38},3.4\cdot10^{38}]$	Hasta 6 dígitos significativos
double	Doble	64bits	$[-1.79\cdot10^{308},1.79\cdot10^{308}]$	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 simultaneamente.

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**

cheatography.com/alejandrolara/

Published 4th March, 2024.
Last updated 4th June, 2024.

adc_lpc40xx.h

```
#define SEL_CANAL_0 (1u << 0)
#define SEL_CA NAL_1 (1u << 1)
#define SEL_CA NAL_2 (1u << 2)
#define SEL_CA NAL_3 (1u << 3)
#define SEL_CA NAL_4 (1u << 4)
#define SEL_CA NAL_5 (1u << 5)
#define SEL_CA NAL_6 (1u << 6)
#define SEL_CA NAL_7 (1u << 7)
void adc_in ici ali zar (ui nt32_t frecue nci -
a_adc, uint32_t canales);
uint32_t adc_co nve rti r(u int32_t canal);
float32_t adc_tr adu cir _a_ ten sio n(u int32_t
result ado _adc);
```

glcd.h

```
#define NEGRO RGB(0, 0, 0)
#define AZUL_O SCURO RGB(0, 0, 128)
#define VERDE_ OSCURO RGB(0, 128, 0)
#define CIAN_O SCURO 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_O SCURO 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_i nic ial iza r(v oid);
void glcd_b orr ar( uin t16_t color);
int32_t glcd_x pri ntf (in t32_t x,
                                    int32_t
y,
                                    uin t16_t
color,
                                    uin t16_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)	dato |= (1<<p)
0 en bit *p*	~(1<<p)	dato &= ~(1<<p)
Selec. bit *p*	(dato>>p)&1	

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 linea
/* ..... */	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 utilicemos
#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 necesitemos
   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_ ret ard o_m s(L PC_ TIM _Ty peDef
*timer _regs,
                                          uin -
t32_t retard o_e n_ms);

void timer_ ret ard 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);
```