Manual de primitivas de la Biblioteca de Infotronic

Versión 240618

Lectura del Teclado 4x1:

uint8_t GetKey (void)

Trae del buffer de teclado el código de la tecla pulsada.

Parámetros: void

Retorno: Código de tecla o NO_KEY (0xFF)

(#define NO_KEY 0xFF)

Entradas Digitales (ED):

uint8_t LeerED (uint8_t *nEntrada*)

Lee el valor de la ED solicitada.

Parámetro:

nEntrada: Número de entrada

macros:

ENTRADA0: 0
ENTRADA1: 1
ENTRADA2: 2

Retorno: valor de la entrada

LCD: *Muestra una string en un LCD de 2 x 16.*

void LCD_Display (const char **string*, uint8_t *line*, uint8_t *pos*)

Parámetros:

string	dirección de comienzo de la string
line	numero de renglón del Display (DSP0: renglón superior, DSP1: renglón inferior)
pos	posición relativa dentro del renglón

Retorno: void

Salidas Digitales (Relays):

void Relays (uint8_t *nRelay*, uint8_t *estado*)

Activa o desactiva un relay y su led asociado.

Parámetros:

nRelay: Numero de relay
estado: ON (1) u OFF(0)

Números de relays en las siguientes macros

RELAY0: 0 - RELAY1: 1
RELAY2: 2 - RELAY3: 3

Retorno: void

Salidas Digitales (RGB):

void LedsRGB (uint8_t *led*, uint8_t *estado*)

Activa o desactiva uno de los leds del RGB.

Parámetros:

- *led*: usar las siguientes macros

  ROJO: 0
  VERDE: 1
  AZUL: 2

- *estado*: ON (1) u OFF(0)

Retorno: void

Salidas Digitales (buzzer):

void Buzzer (uint8_t *estado*)

Activa o desactiva el buzzer.

Parámetros:

- *estado*: ON (1) u OFF(0)

Retorno: void

Display7seg:

void Display (unsigned int *val*, unsigned char *dsp*)

Muestra el valor que recibe a través de val en el display indicado por dsp. Los seis dígitos de Infotronic constituyen dos displays de 3 dígitos (DSP0 = 0 y DSP1 = 1)

Parámetros

val: Valor a mostrar en el display elegido
dsp: Display elegido para mostrar el valor Val

Retorno: void

Timers: Arrancar un timer de los 32 disponibles

void TimerStart (uint8_t *event*, timer_t *t*, void (**handler*)(void) , uint8_t base)

*Inicia el timer identificado por **event** y al transcurrir el tiempo especificado por **t** y **base**, se llama a la función apuntada por **handler**.*

Parámetros:

- *event*: Numero de evento entre 0 y 31
- *t*: Tiempo del evento. Dependiente de la base de tiempos
- *handler*: Rutina que atiende el evento a su vencimiento.
- *base*: Base de tiempo elegida (DEC=decimas - SEG=segundos - MIN=minutos)

Retorno: void

Timers: *Reinicia el timer del evento **event** con el valor **t** (no lo resetea)*

void SetTimer (uint8_t *event* , timer_t *t*)

Parámetros:

- *event*: Numero de evento entre 0 y 31
- *t*: Tiempo del evento. (depende de la base de tiempos)

Retorno: void

Timers: *Lee el valor al vuelo del timer del evento **event**.*

uint32_t GetTimer (uint8_t *event*)

Parámetros: event: Numero de evento entre 0 y 31

Retorno: valor del timer

Timers: *Detiene/Arranca el timer. NO lo resetea. Lo pone o lo saca de stand by*

void StandByTimer (uint8_t *event* , uint8_t *accion*)

Parámetros:

- *event*: Numero de evento entre 0 y 31
- *acción*: RUN lo arranca, PAUSE lo pone en stand by

Retorno: void

Timers: *Detiene TODOS los timers activos.*

void Timer_Close (void)

Parámetros: void

Retorno: void

Timers: *Detiene el timer **event**.*

void Timer_Stop (uint8_t *event*)

Parámetro: Numero de evento entre 0 y 31

Retorno: void

UART: Transmitir

int16_t Transmitir (uint8_t *com* , const void* *datos* , uint8_t *cant*)

Despacha los datos a transmitir

Parámetros:

- *com*: Puerto que será utilizado [UART0 o UART1]
- *datos*: puntero a los datos a transmitir
- *cant*: cantidad de datos a transmitir

Retorno:

- 0 por éxito
- -1 por Error (datos excedidos)

UART: Recibir caracter vía alguna de las 2 UARTs disponibles

int16_t Recibir (uint_8 uart) [MACRO]

Recibe un carácter de la UARTx

Es una **macro** de la función: **int16_t Recibir (uint_8t uart)**

```
#define  Recibir( x )  (( x == UART0 )?( UART0_PopRX( void )):(UART1_PopRX( void )))
```

Parámetros: identificación de la UART a usar (0 = UART0 ó 1 = UART1)

Retorno:

- Carácter recibido
- -1 por Error (no hay datos para leer)

Adc: lectura del potenciómetro

int16_t Potenciometro (void)

Retorna el valor de tensión asociado al potenciómetro.

Parámetros: void

Retorno: Lectura del pote (rango 0 a 3.3V)

Adc: lectura del termistor

int16_t Temperatura (void)

Retorna el valor de temperatura del termistor

Parámetros: void

Retorno: temperatura (rango -50 a 120)

Adc: lectura de la bornera asociada al ADC

int16_t ADC_Externa (void)

Retorna el valor de tensión asociado a la bornera.

Parámetro: void

Retorno: Lectura de cuentas

Tipos de datos:

```
typedef  unsigned int          uint32_t;
typedef  short unsigned int    uint16_t;
typedef  unsigned char         uint8_t ;
typedef  int                   int32_t;
typedef  short int             int16_t;
typedef  char                  int8_t;
```

```
typedef void (*Timer_Handler)(void);
```