Manual de primitivas de la Biblioteca de Infotronic

Inicialización del Kit: Inicializa la placa, muy importante invocarla siempre.

```
void Inicializacion (void)
```

ENTRADAS DIGITALES

Lectura del Teclado 4x1: Trae del buffer de teclado el código de la tecla pulsada.

```
uint8_t GetKey (void)
```

Retorno: Código de tecla entre 0 y 4 o NO_KEY (0xFF) si no hay tecla presionada.

Nota: Una vez devuelto un valor de tecla, no vuelve a devolverlo hasta que la misma se suelte y se vuelva a presionar.

Entradas Digitales (ED): Lee el valor de la ED solicitada.

```
uint8_t Entradas (uint8_t nEntrada)
```

nEntrada: Número de entrada macros:
ENTRADA0 (0) - ENTRADA1 (1) - ENTRADA2 (2)

Retorno: valor de la entrada

SALIDAS DIGITALES

Salidas Digitales (Relays): Activa o desactiva un relay y su led asociado.

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

nRelay: Numero de relay
RELAY0 (0) - RELAY1 (1) - RELAY2 (2) - RELAY3 (3)

estado: ON (1) u OFF(0)

Led RGB: Activa o desactiva uno de los leds del RGB.

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

led: ROJO (1) - VERDE (2) - AZUL (0)

estado: ON (1) u OFF(0)

Buzzer: Activa o desactiva el buzzer

```
void Buzzer (uint8_t estado)
```

estado: ON (1) u OFF(0)

VISUALIZACIÓN

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

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

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

Display7seg: 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.

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

val: Valor a mostrar en el display elegido
dsp: Display elegido para mostrar el valor Val, DSP0 (0) y DSP1 (1)

TEMPORIZACIÓN

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

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

event: Número 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=décimas - SEG=segundos - MIN=minutos)

Reiniciar el Timer: Reinicia el timer del evento event con el valor t (no lo resetea)

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

event: Número de evento entre 0 y 31
t: Tiempo del evento. Dependiente de la base de tiempos

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

```
uint32_t GetTimer (uint8_t event)
```

event: Número de evento entre 0 y 31

Pausar/iniciar el Timer: Detiene/Arranca el timer. NO lo resetea. Lo pone o lo saca de stand by

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

event: Número de evento entre 0 y 31
acción: RUN lo arranca, PAUSE lo pone en stand by.

Detener Todos: Detiene TODOS los timers activos.

```
void TimerClose (void)
```

Detener uno: Detiene el timer event.

```
void TimerStop (uint8_t event)
```

event: Número de evento entre 0 y 31

Analiza timers: Analiza timers y dispara el handler declarado para cada timer vencido. Debe ubicarse en el while(1).

```
void TimerEvent(void);
```

COMUNICACIÓN SERIE (UART)

Transmitir: Despacha los datos a transmitir

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

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)

Recibir: Recibe UN caracter de la UARTx

```
int16_t Recibir (uint_8 uart)
```

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

Retorno: Carácter recibido o (-1) cuando no hay datos para leer.

CONVERSOR ANALÓGICO DIGITAL (ADC)

Lectura del potenciómetro: Retorna el valor de tensión asociado al potenciómetro.

```
int16_t Potenciometro ( void )
```

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

Lectura del termistor: Retorna el valor de temperatura del termistor

```
int16_t Temperatura ( void )
```

Retorno: temperatura (rango -50 a 120)

Lectura Adc: Retorna el valor de tensión asociado a la bornera del ADC

```
int16_t ADC_Externa ( 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;
```