KITLPC845-C++

2.0

Generated by Doxygen 1.13.1

Chapter 1

Topic Index

1.1 Topics

Here is a list of all topics with brief descriptions:

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Topic Documentation

5.1 Drivers

Classes

- class teclado

  Clase del objeto teclado FUNCIONAMIENTO: Este objeto permite controlar teclados matriciales cableados, eliminando el rebote mecánico. Teclado de tipo mono-usuario, mono-dedo con opción de mantener presionado una tecla.
- class I4017

  Clase del objeto I4017 El objeto I4017 permite el control del integrado del mismo nombre. Habitualmente utilizado para barrer información a tavés de sus patas.
- class I4511

  Clase del objeto I4511 El objeto I4511 permite el control del integrado del mismo nombre. Este integrado permite el control de un display 7 segmentos mediante una comunicación binaria en formato paralelo.
- class digito

  Clase del objeto digito El objeto digito posee todas las funcionalidades y propiedades de forma que pueda ser la representación en código de un dígito. Un ejemplo de esto sería un display de 7 segmentos.
- class display7Segmentos

  Clase del objeto display7Segmentos El objeto display7Segmentos permite el control de un display con dígitos de 7 segmentos agrupados y controlados con un integrado de barrido. Para su funcionamiento, utiliza el systick y escribe de un led a la vez a altas velocidades. La velocidad de escritura depende de la frecuencia del systick y del valor asignado a m_ticks. Para ver mejores resultados modificar dicho valor.
- struct gruposdedigitos

  Estructura de grupo de dígitos.
- class LCD

  Clase del objeto lcd El objeto lcd permite el manejo de displays digitales mediante comunicación de 4 bits.
- class L298N

  Clase del objeto L298N El objeto L298N realiza las acciones de control de dos motores controlados por el correspondiente periférico.
- class Reloj

  Clase del objeto Reloj.
- class Pwm

  Clase del objeto Pwm.
- class PWM_Reader

  Clase del objeto PWM_Reader.
- class HC_SR04

Clase del objeto HC_SR04 El objeto HC_SR04 Mide distancia mediante el uso de un ultrasónico. Debido a los tiempos muy pequeños de uso, no se recomienda utilizar en grandes cantidades.

- class Pin

  Clase del objeto Pin.
- class gpio

  Clase del objeto gpio.
- class PinInterrupt

  Clase del objeto Pin_interrupt El objeto Pin_interrupt debe ser heredado por cualquier objeto que desee tener interupciones por pin.
- class Input

  Clase del objeto Input.
- class Output

  Clase del objeto outputs.
- class Callback

  Clase del objeto Callback.
- class Timer

  Clase del objeto timer.
- class timers

  Clase del objeto timers El objeto timers permite agrupar todos los timers y ejecutarlos de una sola pasada. Permite ahorrar código.
- class MRThandler

  Clase del objeto MRThandler El objeto MRThandler debe ser heredado por cualquier objeto que desee estar conectado a las interrupciones del MRT timer.
- class SCtimer

  Clase del objeto SCtimer El objeto SCtimer debe ser heredado por quienes desen utilizar las interrucpiones o funcionalidades del SCtimer.
- class DAC

  Clase del objeto DAC FUNCIONAMIENTO: Realiza una conversion digital->analógica en un rango desde 0 hasta max_range. El rango real del LPC845 va de 0 hasta 1023. Se realiza una conversion lineal entre el rango del dispositivo y el utilizado por el usuario. NO USAR EL CHANNEL 1. El canal existe segun datasheet pero los registros son vagos y poco explicativos. Corresponde al PINENABLE. Recomendado utilizar solo el CHANNEL 0.
- class ADC

  Clase del objeto ADC FUNCIONAMIENTO: La clase ADC utiliza el ADCGroup para poder ser manejada de forma individual por cada pata. Se pueden crear tantos objetos como canales del ADC existen. La configuración de conversión se realiza automáticamente con el primer objeto ADC creado, el resto no necesita recibir ninguna frecuencia de clock o muestreo.
- class ADC_Group

  Clase del objeto ADC_Group FUNCIONAMIENTO: Solo debe crearse UN objeto ADC. Todos los canales y distintos ADC son manejados por el mismo objeto. Realiza un barrido y guarda a todos los ADC en un vector de resultados.
- class Uart

  Clase del objeto uart El objeto uart genera una comunicación asincrónica de tipo UART.
- class I2C

  Clase del objeto I2C El objeto I2C genera una comunicación sincrónica de tipo I2C. Posee las funciones basicas como start, stop, write y read.
- class I2CMaster

  Clase del objeto I2CMaster El objeto I2CMaster genera una comunicación tipo master de I2C utilizando buffers de recepcion y transmision con interrupciones.
- class SPI

Typedefs

- typedef void(∗ Timer_Handler) (void)

Variables

- const uint8_t IOCON_INDEX_PIO0 [ ] = { 17,11,6,5,4,3,16,15,4,13,8,7,2,1,18,10,9,0,30,29,28,27,26,25,24,23,22,21,20,0,0,35}
- const uint8_t IOCON_INDEX_PIO1 [ ] = { 36,37,3,41,42,43,46,49,31,32,55,54,33,34,39,40,44,45,47,48,52,53,0,0,0,0,0,0,0,50,5
- PinInterrupt * g_gpiohandler [MAX_PININTERRUPT]
- vector< Callback * > g_Handler
- std::vector< MRThandler * > g_MRThandler

5.1.1 Detailed Description

USO: Se crea y se inicializa. Posee 2 modos: Master y Slave. MASTER: Start(): Inicia la comunicacion con Start en accion de read/write. Solo puede llamarse a start si el I2C está en idle o pendiente de lectura.

Read() y Write(): Se escribe y se lee de a 1 byte con las funciones Read y Write. Previamente se debe hacer el start correspondiente para funcionar. Solo puede llamarse a Read y Write si el I2C esta en tx_ready o rx_data.

Stop(): Al finalizar se le da a stop y se termina la conversación con el dispositivo. Solo puede llamarse a Stop si el I2C esta en modos distintos al busy.

SLAVE: El slave nunca inicia la comunicacion. Debe revisar si el mismo se encuentra pendiente (estados slvst_↩ addr, slvst_tx o slvst_rx)

ACKAddr(): Para iniciar la comunicación, se debe reconocer que el address traido es el correspondiente al dispositivo.

Read(): Si llegó algo al slave, se leera. Solo funcionara si el I2C esta en slvst_rx. Debe realizarse un ACK() para continuar.

Write(): Si llego una peticion de escritura al slave, se escribe. Solo funcionara si el I2C esta en slvst_tx.

ACK(): Cuando se recibe un dato, se debe reconocer la llegada del mismo.

USO: Se crea y se inicializa. Se envia informacion con Write() y se pide lectura con RequestRead(). Solo se pueden ejecutar si el I2C esta en idle. Write(): Escribe el string en el buffer y lo envia continuamente por interrupcion. RequestRead(): Pide una lectura continua de n bytes que se guardan en un buffer por interrupcion. Read(): Lee el buffer de recepcion. Puede leerse en cualquier momento y por partes. isIdle(): Indica si el I2C esta en reposo. Solo cuando esto sea cierto se podra leer y escribir al slave.

Aun no soporta timeout, ni NACKs recibidos. I2C posee los metodos para agregar la funcion de NACK.

USO: Se crea y se inicializa. Posee 2 modos: Master y Slave. MASTER: Start(): Inicia la comunicacion con Start en accion de read/write. Solo puede llamarse a start si el I2C está en idle o pendiente de lectura.

Read() y Write(): Se escribe y se lee de a 1 byte con las funciones Read y Write. Previamente se debe hacer el start correspondiente para funcionar. Solo puede llamarse a Read y Write si el I2C esta en tx_ready o rx_data.

Stop(): Al finalizar se le da a stop y se termina la conversación con el dispositivo. Solo puede llamarse a Stop si el I2C esta en modos distintos al busy.

SLAVE: El slave nunca inicia la comunicacion. Debe revisar si el mismo se encuentra pendiente (estados slvst_↩ addr, slvst_tx o slvst_rx)

ACKAddr(): Para iniciar la comunicación, se debe reconocer que el address traido es el correspondiente al dispositivo.

Read(): Si llegó algo al slave, se leera. Solo funcionara si el I2C esta en slvst_rx. Debe realizarse un ACK() para continuar.

Write(): Si llego una peticion de escritura al slave, se escribe. Solo funcionara si el I2C esta en slvst_tx.

ACK(): Cuando se recibe un dato, se debe reconocer la llegada del mismo.

5.1.2 Typedef Documentation

5.1.2.1 Timer_Handler

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

Tipo de dato: funcion a ejecutar al terminar el timer.

5.1.3 Variable Documentation

5.1.3.1 g_gpiohandler

```
PinInterrupt* g_gpiohandler[MAX_PININTERRUPT]  [extern]
```

Vector de interrupciones del PIN INTERRUPT

5.1.3.2 g_Handler

```
vector<Callback*> g_Handler  [extern]
```

Vector de interrupciones del systick

5.1.3.3 g_MRThandler

```
std::vector<MRThandler *> g_MRThandler  [extern]
```

Vector de interrupciones del MRT Timer

5.1.3.4 IOCON_INDEX_PIO0

```
const uint8_t IOCON_INDEX_PIO0[] = { 17,11,6,5,4,3,16,15,4,13,8,7,2,1,18,10,9,0,30,29,28,27,26,25,24,23,22,21,
```

Index for the IOCON Register Port0

5.1.3.5 IOCON_INDEX_PIO1

```
const uint8_t IOCON_INDEX_PIO1[] = { 36,37,3,41,42,43,46,49,31,32,55,54,33,34,39,40,44,45,47,48,52,53,0,0,0,0,
```

Index for the IOCON Register Port1

5.2 Abstracta

Classes

- class barrido

  Clase del objeto barrido Clase abstracta pura para la generación de barridos.
- class segmentos

  Clase del objeto segmentos Clase abstracta pura para la generación de segmentos.
- class Display

  Clase del objeto display Clase abstracta pura para la generación de displays.
- class Puente_H

  Clase del objeto Puente_H El objeto Puente_H es la interfaz abstracta pura de cualquier puente H que se desee realizar.
- class distancia

  Clase del objeto distancia Clase abstracta pura para la generación de HCS-R04.
- class InOut

  Clase del objeto InOut.
- class ComunicacionAsincronica

  Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de UART.
- class ComunicacionSincronica

  Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de comunicaciones sincrónicas como la I2C o la SPI.

5.2.1 Detailed Description

Chapter 6

Class Documentation

6.1 ADC Class Reference

Clase del objeto ADC FUNCIONAMIENTO: La clase ADC utiliza el ADCGroup para poder ser manejada de forma individual por cada pata. Se pueden crear tantos objetos como canales del ADC existen. La configuración de conversión se realiza automáticamente con el primer objeto ADC creado, el resto no necesita recibir ninguna frecuencia de clock o muestreo.

`#include <ADC.h>`

Inheritance diagram for ADC:

Collaboration diagram for ADC:

Public Member Functions

- ADC (uint8_t _channel, uint32_t _clk_freq=0, uint32_t _sample_rate=0)

  Default constructor of an ADC.
- int32_t Get (void)

  Devuelve el valor del ADC.
- bool IsResultReady (void)

  Indica si la conversión análoga->digital fue terminada.
- void Trigger (void)

  Ejecuta una conversión analógica->digital.
- void Inicializar (void)

  Inicializador del ADC.

Public Member Functions inherited from Pin

- Pin (port_t port, uint8_t bit)

  Constructor de clase PIN.

Additional Inherited Members

Public Types inherited from Pin

- enum port_t { **port0** , **port1** }
- enum max_bits_port_t { **b_port0** = 31 , **b_port1** = 9 }

  máximos pines por puerto
- enum error_t { **error** = 2 , **ok** }
- typedef enum Pin::port_t port_t
- typedef enum Pin::error_t error_t

Public Attributes inherited from Pin

- const port_t m_port
- const uint8_t m_bit
- int8_t m_error

6.1.1 Detailed Description

Clase del objeto ADC FUNCIONAMIENTO: La clase ADC utiliza el ADCGroup para poder ser manejada de forma individual por cada pata. Se pueden crear tantos objetos como canales del ADC existen. La configuración de conversión se realiza automáticamente con el primer objeto ADC creado, el resto no necesita recibir ninguna frecuencia de clock o muestreo.

6.1.2 Constructor & Destructor Documentation

6.1.2.1 ADC()

```
ADC::ADC (
            uint8_t _channel,
            uint32_t _clk_freq = 0,
            uint32_t _sample_rate = 0)
```

Default constructor of an ADC.

Crea un ADC en el canal correspondiente. La frecuencia y muestreo solo son necesarios de colocar en el primer ADC construido ya que el periférico utiliza todos con la misma configuracion

Parameters

in	*_channel*	Canal del ADC.
in	*_clk_freq*	Frecuencia de clock del periférico ADC.
in	*_sample_rate*	Frecuencia de muestreo del periférico ADC.

6.1.3 Member Function Documentation

6.1.3.1 Get()

```
int32_t ADC::Get (
            void )
```

Devuelve el valor del ADC.

Returns

Valor convertido del ADC.

6.1.3.2 Inicializar()

```
void ADC::Inicializar (
            void )
```

Inicializador del ADC.

Agrega el canal al barrido de lecturas del ADC.

6.1.3.3 IsResultReady()

```
bool ADC::IsResultReady (
            void )
```

Indica si la conversión análoga->digital fue terminada.

Returns

bool: verdadero si el resultado se encuentra, falso sino.

6.1.3.4 Trigger()

```
void ADC::Trigger (
            void )
```

Ejecuta una conversión analógica->digital.

El ADC empieza la conversión de todos los canales activos.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/12-ADC/ADC.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/12-ADC/ADC.cpp

6.2 ADC_Group Class Reference

Clase del objeto ADC_Group FUNCIONAMIENTO: Solo debe crearse UN objeto ADC. Todos los canales y distintos ADC son manejados por el mismo objeto. Realiza un barrido y guarda a todos los ADC en un vector de resultados.

`#include <ADCGroup.h>`

Public Types

- enum adc_isr { **SEQA_ISR** , **SEQB_ISR** , **THCMP_ISR** , **OVR_ISR** }
- enum irq_source_inten { **ADC_SEQA_IRQ_INTEN** = 0 , **ADC_SEQB_IRQ_INTEN** = 1 , **ADC_OVR_IRQ_INTEN** = 2 , **INVALID_IRQ_INTEN** = 0xFF }
- enum error_t { **OK** = 0 , **ERROR** = -1 }
- typedef enum ADC_Group::adc_isr adc_isr
- typedef enum ADC_Group::irq_source_inten irq_source_inten
- typedef enum ADC_Group::error_t error_t

Public Member Functions

- ADC_Group (uint32_t clk_freq, uint32_t sample_rate, bool init_channel0=false)

  Constructor de clase ADC.
- virtual ~ADC_Group ()=default
- void **Inicializar** (void)

  Inicializa el ADC en la secuencia A.
- void SetLowPowerMode (bool low_power)

  Activa/desactiva el modo de bajo consumo.
- void **SetSampleRate** (void)

  Setea la frecuencia de muestreo.
- void EnableIrq (irq_source_inten irq)

  Habilita la interrupcion.
- void DisableIrq (irq_source_inten irq)

  Deshabilita la interrupcion.
- ADC_Group::error_t InitADCChanel (uint8_t channel)

  Inicializa el canal indicado con la secuencia A.
- ADC_Group::error_t RemoveADCChanel (uint8_t channel)

  Desconfigura el canal indicado de la secuencia A del ADC.
- void **TriggerStartSeqA** (void)

  Inicia la conversión analógica - digital de la secuencia A.
- int32_t GetValue (uint8_t channel)

  Devuelve el valor de conversión guardado en el buffer. NO EL ACTUAL DEL REGISTRO.
- bool IsResultReady (uint8_t channel) const

  Indica si la conversión de channel está terminada o no.
- void Handler (adc_isr isr)

  Handler del ADC.

6.2.1 Detailed Description

Clase del objeto ADC_Group FUNCIONAMIENTO: Solo debe crearse UN objeto ADC. Todos los canales y distintos ADC son manejados por el mismo objeto. Realiza un barrido y guarda a todos los ADC en un vector de resultados.

6.2.2 Member Typedef Documentation

6.2.2.1 adc_isr

```
typedef enum ADC_Group::adc_isr ADC_Group::adc_isr
```

Tipo de interrupciones del ADC

6.2.2.2 error_t

```
typedef enum ADC_Group::error_t ADC_Group::error_t
```

Error en la clase ADC

6.2.2.3 irq_source_inten

```
typedef enum ADC_Group::irq_source_inten ADC_Group::irq_source_inten
```

Fuente de la interrupcion del ADC

6.2.3 Member Enumeration Documentation

6.2.3.1 adc_isr

```
enum ADC_Group::adc_isr
```

Tipo de interrupciones del ADC

6.2.3.2 error_t

```
enum ADC_Group::error_t
```

Error en la clase ADC

6.2.3.3 irq_source_inten

```
enum ADC_Group::irq_source_inten
```

Fuente de la interrupcion del ADC

6.2.4 Constructor & Destructor Documentation

6.2.4.1 ADC_Group()

```
ADC_Group::ADC_Group (
            uint32_t clk_freq,
            uint32_t sample_rate,
            bool init_channel0 = false)
```

Constructor de clase ADC.

Crea un ADC con los parámetros correspondientes

Parameters

in	*clk_freq*	Frecuencia del periférico ADC.
in	*sample_rate*	Frecuencia de muestreo del ADC.
in	*init_channel0*	Si se inicia el canal 0 o no.

6.2.4.2 ~ADC_Group()

```
virtual ADC_Group::~ADC_Group ()  [virtual], [default]
```

Destructor por defecto

6.2.5 Member Function Documentation

6.2.5.1 DisableIrq()

```
void ADC_Group::DisableIrq (
            irq_source_inten irq)
```

Deshabilita la interrupcion.

Parameters

in	*irq*	Interrupcion a deshabilitar

6.2.5.2 EnableIrq()

```
void ADC_Group::EnableIrq (
            irq_source_inten irq)
```

Habilita la interrupcion.

Parameters

in	*irq*	Interrupcion a habilitar

6.2.5.3 GetValue()

```
int32_t ADC_Group::GetValue (
            uint8_t channel)
```

Devuelve el valor de conversión guardado en el buffer. NO EL ACTUAL DEL REGISTRO.

Parameters

in	*channel*	Canal a obtener la conversión

Returns

valor de la conversión

6.2.5.4 Handler()

```
void ADC_Group::Handler (
            adc_isr isr)
```

Handler del ADC.

Funcion Handler de todas las interrupciones posibles del ADC.

Parameters

in	*isr*	Tipo de interrupción

6.2.5.5 InitADCChanel()

```
ADC_Group::error_t ADC_Group::InitADCChanel (
            uint8_t channel)
```

Inicializa el canal indicado con la secuencia A.

Parameters

in	*channel*	Canal a inicializar

Returns

mensaje de error

6.2.5.6 IsResultReady()

```
bool ADC_Group::IsResultReady (
            uint8_t channel) const
```

Indica si la conversión de channel está terminada o no.

Parameters

in	*channel*	Canal a preguntar.

Returns

verdadero = conversión lista. falso = conversión en proceso

6.2.5.7 RemoveADCChanel()

```
ADC_Group::error_t ADC_Group::RemoveADCChanel (
            uint8_t channel)
```

Desconfigura el canal indicado de la secuencia A del ADC.

Parameters

in	*channel*	Canal a desconfigurar

Returns

mensaje de error

6.2.5.8 SetLowPowerMode()

```
void ADC_Group::SetLowPowerMode (
            bool low_power)
```

Activa/desactiva el modo de bajo consumo.

Configura el ADC para no consumir energia al no realizar acciones o sí consumir.

Parameters

in	*low_power*	Valor de seteo del bajo consumo

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/12-ADC/ADCGroup.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/12-ADC/ADCGroup.cpp

6.3 barrido Class Reference

Clase del objeto barrido Clase abstracta pura para la generación de barridos.

```
#include <Barrido.h>
```

Inheritance diagram for barrido:

Public Member Functions

- virtual void SetDigito (void)=0
- virtual void Inicializar (void)=0

6.3.1 Detailed Description

Clase del objeto barrido Clase abstracta pura para la generación de barridos.

6.3.2 Member Function Documentation

6.3.2.1 Inicializar()

```
virtual void barrido::Inicializar (
            void )  [pure virtual]
```

Funcion de inicializacion

Implemented in I4017.

6.3.2.2 SetDigito()

```
virtual void barrido::SetDigito (
            void )  [pure virtual]
```

constructor por defecto Funcion de encendido del barrido

Implemented in I4017.

The documentation for this class was generated from the following file:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/A-↩ Barrido/Barrido.h

6.4 Callback Class Reference

Clase del objeto Callback.

```
#include <Callback.h>
```

Inheritance diagram for Callback:

Public Member Functions

- void SetInterrupt ()
  Activa la interrupción.
- void UnSetInterrupt ()
  Desactiva la interrupción.
- virtual void SWhandler (void)=0

6.4.1 Detailed Description

Clase del objeto Callback.

El objeto Callback debe ser heredado y otorga a los herederos la conexión al systick y a la interrupción temporizada

6.4.2 Member Function Documentation

6.4.2.1 SetInterrupt()

```
void Callback::SetInterrupt (
            void )
```

Activa la interrupción.

Coloca al objeto dentro de la lista del handler systick.

6.4.2.2 SWhandler()

```
virtual void Callback::SWhandler (
            void )  [pure virtual]
```

Este método debe ser implementado por las clases derivadas cada una resolverá que hacer con su irq enganchada al systick del sistema

Implemented in display7Segmentos, Input, LCD, Output, Reloj, teclado, and Timer.

6.4.2.3 UnSetInterrupt()

```
void Callback::UnSetInterrupt (
            void )
```

Desactiva la interrupción.

Borra de la lista handler al objeto actual.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/06-Callback/Callback.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/06-Callback/Callback.cpp

6.5 ComunicacionAsincronica Class Reference

Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de UART.

```
#include <ComunicacionAsincronica.h>
```

Inheritance diagram for ComunicacionAsincronica:

Public Member Functions

- ComunicacionAsincronica ()=default
- virtual void Transmit (const char *msg)=0
- virtual void Transmit (const void *msg, uint32_t n)=0
- virtual void * Message (void *msg, uint32_t n)=0
- virtual void UART_IRQHandler (void)=0
- virtual ~ComunicacionAsincronica ()=default

Protected Member Functions

- virtual void pushRx (uint8_t dato)=0
- virtual uint8_t popRx (uint8_t *dato)=0
- virtual void pushTx (uint8_t dato)=0
- virtual uint8_t popTx (uint8_t *dato)=0

6.5.1 Detailed Description

Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de UART.

6.5.2 Constructor & Destructor Documentation

6.5.2.1 ComunicacionAsincronica()

```
ComunicacionAsincronica::ComunicacionAsincronica ()  [default]
```

Constructor por defecto

6.5.2.2 ~ComunicacionAsincronica()

```
virtual ComunicacionAsincronica::~ComunicacionAsincronica ()  [virtual], [default]
```

Destructor por defecto

6.5.3 Member Function Documentation

6.5.3.1 Message()

```
virtual void * ComunicacionAsincronica::Message (
            void * msg,
            uint32_t n)  [pure virtual]
```

Funcion de lectura

Implemented in ESP8266, and Uart.

6.5.3.2 popRx()

```
virtual uint8_t ComunicacionAsincronica::popRx (
            uint8_t * dato)  [protected], [pure virtual]
```

Devuelve recepcion

6.5.3.3 popTx()

```
virtual uint8_t ComunicacionAsincronica::popTx (
            uint8_t * dato)  [protected], [pure virtual]
```

Devuelve transmision

6.5.3.4 pushRx()

```
virtual void ComunicacionAsincronica::pushRx (
            uint8_t dato)  [protected], [pure virtual]
```

Envia recepcion

6.5.3.5 pushTx()

```
virtual void ComunicacionAsincronica::pushTx (
            uint8_t dato)  [protected], [pure virtual]
```

Envia transmision

6.5.3.6 Transmit() [1/2]

```
virtual void ComunicacionAsincronica::Transmit (
            const char * msg)  [pure virtual]
```

Funcion de transmitir

Implemented in ESP8266, and Uart.

6.5.3.7 Transmit() [2/2]

```
virtual void ComunicacionAsincronica::Transmit (
            const void * msg,
            uint32_t n)  [pure virtual]
```

Funcion de transmitir

Implemented in ESP8266, and Uart.

6.5.3.8 UART_IRQHandler()

```
virtual void ComunicacionAsincronica::UART_IRQHandler (
            void )  [pure virtual]
```

Funcion de interrupcion

The documentation for this class was generated from the following file:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/13-UART/ComunicacionAsincronica.h

6.6 ComunicacionSincronica Class Reference

Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de comunicaciones sincrónicas como la I2C o la SPI.

`#include <ComunicacionSincronica.h>`

Inheritance diagram for ComunicacionSincronica:

Collaboration diagram for ComunicacionSincronica:

Public Member Functions

- ComunicacionSincronica ()=default
- virtual void Write (uint8_t data)=0
- virtual ~ComunicacionSincronica ()=default

Protected Attributes

- const Pin * m_scl

6.6.1 Detailed Description

Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de comunicaciones sincrónicas como la I2C o la SPI.

6.6.2 Constructor & Destructor Documentation

6.6.2.1 ComunicacionSincronica()

```
ComunicacionSincronica::ComunicacionSincronica ()  [default]
```

Constructor por defecto

6.6.2.2 ~ComunicacionSincronica()

```
virtual ComunicacionSincronica::~ComunicacionSincronica ()  [virtual], [default]
```

Destructor por defecto

6.6.3 Member Function Documentation

6.6.3.1 Write()

```
virtual void ComunicacionSincronica::Write (
            uint8_t data)  [pure virtual]
```

Funcion de escritura

Implemented in I2C, and SPI.

6.6.4 Member Data Documentation

6.6.4.1 m_scl

```
const Pin* ComunicacionSincronica::m_scl  [protected]
```

Pin de Clock. Debe existir en toda comunicacion sincronica

The documentation for this class was generated from the following file:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/14-ComSincronica/ComunicacionSincronica.h

6.7 DAC Class Reference

Clase del objeto DAC FUNCIONAMIENTO: Realiza una conversion digital->analógica en un rango desde 0 hasta max_range. El rango real del LPC845 va de 0 hasta 1023. Se realiza una conversion lineal entre el rango del dispositivo y el utilizado por el usuario. NO USAR EL CHANNEL 1. El canal existe segun datasheet pero los registros son vagos y poco explicativos. Corresponde al PINENABLE. Recomendado utilizar solo el CHANNEL 0.

```
#include <DAC.h>
```

Inheritance diagram for DAC:

Collaboration diagram for DAC:

Public Types

- enum dac_channel { **DAC_CHANNEL_0** = 0 , **DAC_CHANNEL_1** = 1 }
- enum dac_error { **ERROR** = -1 , **OK** = 0 }
- typedef enum DAC::dac_channel dac_channel
- typedef enum DAC::dac_error dac_error

Public Member Functions

- DAC (dac_channel channel, uint32_t max_range=MAX_DAC_VALUE)

  Constructor de clase DAC.
- DAC::dac_error Inicializar (void)

  Inicializa el DAC.
- void Set (uint32_t val)

  Setea el valor del DAC.
- uint32_t Get (void) const

  Devuelve el valor analógico del DAC.
- void SetMaxRange (uint32_t max_range)

  Setea el rango máximo.
- uint32_t GetMaxRange (void) const

  Devuelve el rango máximo que posee el DAC.
- DAC & operator= (uint32_t val)

  Sobrecarga del operador =.
- bool operator== (uint32_t val) const

Sobrecarga del operador ==.

- bool operator< (uint32_t val) const

  Sobrecarga del operador <.
- bool operator<= (uint32_t val) const

  Sobrecarga del operador <=.
- bool operator> (uint32_t val) const

  Sobrecarga del operador >
- bool operator>= (uint32_t val) const

  Sobrecarga del operador >=.
- bool operator!= (uint32_t val) const

  Sobrecarga del operador !=.
- virtual ~DAC ()

Additional Inherited Members

Protected Types inherited from Pin

- enum port_t { **port0** , **port1** }
- enum max_bits_port_t { **b_port0** = 31 , **b_port1** = 9 }

  máximos pines por puerto
- enum error_t { **error** = 2 , **ok** }
- typedef enum Pin::port_t port_t
- typedef enum Pin::error_t error_t

Protected Member Functions inherited from Pin

- Pin (port_t port, uint8_t bit)

  Constructor de clase PIN.

Protected Attributes inherited from Pin

- const port_t m_port
- const uint8_t m_bit
- int8_t m_error

6.7.1 Detailed Description

Clase del objeto DAC FUNCIONAMIENTO: Realiza una conversion digital->analógica en un rango desde 0 hasta max_range. El rango real del LPC845 va de 0 hasta 1023. Se realiza una conversion lineal entre el rango del dispositivo y el utilizado por el usuario. NO USAR EL CHANNEL 1. El canal existe segun datasheet pero los registros son vagos y poco explicativos. Corresponde al PINENABLE. Recomendado utilizar solo el CHANNEL 0.

6.7.2 Member Typedef Documentation

6.7.2.1 dac_channel

```
typedef enum DAC::dac_channel DAC::dac_channel
```

Canales del DAC

6.7.2.2 dac_error

```
typedef enum DAC::dac_error DAC::dac_error
```

Error en la clase DAC

6.7.3 Member Enumeration Documentation

6.7.3.1 dac_channel

```
enum DAC::dac_channel
```

Canales del DAC

6.7.3.2 dac_error

```
enum DAC::dac_error
```

Error en la clase DAC

6.7.4 Constructor & Destructor Documentation

6.7.4.1 DAC()

```
DAC::DAC (
            dac_channel channel,
            uint32_t max_range = MAX_DAC_VALUE)
```

Constructor de clase DAC.

Crea un DAC con los parámetros correspondientes

Parameters

in	*channel*	Canal del DAC a utilizar.
in	*max_range*	Rango máximo del usuario.

6.7.4.2 ~DAC()

```
DAC::~DAC ()  [virtual]
```

Destructor por defecto

6.7.5 Member Function Documentation

6.7.5.1 Get()

```
uint32_t DAC::Get (
            void ) const
```

Devuelve el valor analógico del DAC.

Devuelve el valor de la salida DAC de acuerdo al rango del usuario

Returns

valor del DAC

6.7.5.2 GetMaxRange()

```
uint32_t DAC::GetMaxRange (
            void ) const
```

Devuelve el rango máximo que posee el DAC.

Devuelve el rango que el usuario se asignó para trabajar

Returns

rango máximo

6.7.5.3 Inicializar()

```
DAC::dac_error DAC::Inicializar (
            void )
```

Inicializa el DAC.

Configura todos los registros de dicho DAC

Returns

Código de error

6.7.5.4 operator"!=()

```
bool DAC::operator!= (
            uint32_t val) const
```

Sobrecarga del operador !=.

Devuelve verdadero o falso si el DAC es distinto a val

Parameters

in	*val*	Valor de comparacion con la salida

Returns

verdadero o falso

6.7.5.5 operator<()

```
bool DAC::operator< (
            uint32_t val) const
```

Sobrecarga del operador <.

Devuelve verdadero o falso si el DAC es menor a val

Parameters

in	*val*	Valor de comparacion con la salida

Returns

verdadero o falso

6.7.5.6 operator<=()

```
bool DAC::operator<= (
            uint32_t val) const
```

Sobrecarga del operador <=.

Devuelve verdadero o falso si el DAC es menor o igual a val

Parameters

in	*val*	Valor de comparacion con la salida

Returns

verdadero o falso

6.7.5.7 operator=()

```
DAC & DAC::operator= (
            uint32_t val)
```

Sobrecarga del operador =.

Fija el valor analógico del DAC a val.

Parameters

in	*val*	Valor de seteo del DAC

Returns

Referencia a si mismo

6.7.5.8 operator==()

```
bool DAC::operator== (
            uint32_t val) const
```

Sobrecarga del operador ==.

Devuelve verdadero o falso si el DAC es igual a val

Parameters

in	*val*	Valor de comparacion con la salida

Returns

verdadero o falso

6.7.5.9 operator>()

```
bool DAC::operator> (
            uint32_t val) const
```

Sobrecarga del operador >

Devuelve verdadero o falso si el DAC es mayor a val

Parameters

in	*val*	Valor de comparacion con la salida

Returns

verdadero o falso

6.7.5.10 operator>=()

```
bool DAC::operator>= (
            uint32_t val) const
```

Sobrecarga del operador >=.

Devuelve verdadero o falso si el DAC es mayor o igual a val

Parameters

in	*val*	Valor de comparacion con la salida

Returns

verdadero o falso

6.7.5.11 Set()

```
void DAC::Set (
            uint32_t val)
```

Setea el valor del DAC.

Enciende la salida analógica con el valor en rango real de voltaje

Parameters

in	*val*	Valor de seteo de la salida

6.7.5.12 SetMaxRange()

```
void DAC::SetMaxRange (
            uint32_t max_range)
```

Setea el rango máximo.

Setea el rango de valores a utilizar por el usuario

Parameters

in	*max_range*	Valor del rango

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/11-DAC/DAC.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/11-DAC/DAC.cpp

6.8 digito Class Reference

Clase del objeto digito El objeto digito posee todas las funcionalidades y propiedades de forma que pueda ser la representación en código de un dígito. Un ejemplo de esto sería un display de 7 segmentos.

```
#include <Digito.h>
```

Public Types

- enum codigo_t { **BCD** , **SEGMENTOS** , **ASCHII** }
- enum modo_t { **APAGAR** = 0xff , **PARPADEAR** = 1 }
- enum SIMBOLOS {
  menos = 10 , **a** , **b** , **c** ,
  d , **e** , **f** , **g** ,
  h , **n** , **o** , **p** ,
  r , **t** , **u** }

Public Member Functions

- digito (codigo_t Sistema=BCD, uint8_t Valor=APAGAR)
- bool Set (uint16_t valor)

  Función de escritura del dígito.
- uint8_t Get (void)

  Devuelve el valor del dígito.
- void Clr (void)

  Función de limpieza del dígito.

6.8.1 Detailed Description

Clase del objeto digito El objeto digito posee todas las funcionalidades y propiedades de forma que pueda ser la representación en código de un dígito. Un ejemplo de esto sería un display de 7 segmentos.

6.8.2 Member Enumeration Documentation

6.8.2.1 codigo_t

```
enum digito::codigo_t
```

Tipos de dígitos

6.8.2.2 modo_t

```
enum digito::modo_t
```

Modos de uso

6.8.2.3 SIMBOLOS

```
enum digito::SIMBOLOS
```

Simbolos a escribir

6.8.3 Constructor & Destructor Documentation

6.8.3.1 digito()

```
digito::digito (
            codigo_t Sistema = BCD,
            uint8_t Valor = APAGAR)  [inline]
```

constructor por defecto

6.8.4 Member Function Documentation

6.8.4.1 Clr()

```
void digito::Clr (
            void )
```

Función de limpieza del dígito.

Deja el dígito apagado del todo. No confundir con escribir un 0, apagado representa sin energía

6.8.4.2 Get()

```
uint8_t digito::Get (
            void )
```

Devuelve el valor del dígito.

Returns

valor del dígito.

6.8.4.3 Set()

```
bool digito::Set (
            uint16_t valor)
```

Función de escritura del dígito.

Escribe el valor del dígito dependiendo de su sistema.

Parameters

in	*valor*	valor a escribir.

Returns

bool. True si tuvo éxito, false sino.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/C-↩ Display/Digito.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/C-↩ Display/Digito.cpp

6.9 Display Class Reference

Clase del objeto display Clase abstracta pura para la generación de displays.

`#include <Display.h>`

Inheritance diagram for Display:

Public Member Functions

- Display ()=default
- virtual void Write (const uint32_t n)=0
- virtual void Clear (void)=0
- virtual ~Display ()=default

6.9.1 Detailed Description

Clase del objeto display Clase abstracta pura para la generación de displays.

6.9.2 Constructor & Destructor Documentation

6.9.2.1 Display()

```
Display::Display ()  [default]
```

Constructor por defecto

6.9.2.2 ~Display()

```
virtual Display::~Display ()  [virtual], [default]
```

destructor por defecto

6.9.3 Member Function Documentation

6.9.3.1 Clear()

```
virtual void Display::Clear (
            void )  [pure virtual]
```

Funcion de limpieza genérica de un display

Implemented in display7Segmentos, and LCD.

6.9.3.2 Write()

```
virtual void Display::Write (
            const uint32_t n)  [pure virtual]
```

Funcion de escritura genérica de un display

Implemented in display7Segmentos.

The documentation for this class was generated from the following file:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/Display.h

6.10 display7Segmentos Class Reference

Clase del objeto display7Segmentos El objeto display7Segmentos permite el control de un display con dígitos de 7 segmentos agrupados y controlados con un integrado de barrido. Para su funcionamiento, utiliza el systick y escribe de un led a la vez a altas velocidades. La velocidad de escritura depende de la frecuencia del systick y del valor asignado a m_ticks. Para ver mejores resultados modificar dicho valor.

```
#include <Display7Segmentos.h>
```

Inheritance diagram for display7Segmentos:

Collaboration diagram for display7Segmentos:

Public Member Functions

- display7Segmentos (vector< gruposdedigitos * > g, segmentos *s, barrido *b, const uint8_t *Posicion↩ Relativa, const digito::codigo_t sistema)

  Objeto del tipo display7Segmentos.
- void SWhandler (void)

  Handler del display de 7 segmentos.
- void Set (uint32_t valor, uint8_t dsp)

  Función de escritura del display de 7 segmentos.
- void Write (const uint32_t n)

  Función de escritura del display de 7 segmentos.
- void Clear (void)

  Función de limpieza del display de 7 segmentos.
- virtual ~display7Segmentos ()

Public Member Functions inherited from Display

- Display ()=default
- virtual ~Display ()=default

6.10.1 Detailed Description

Clase del objeto display7Segmentos El objeto display7Segmentos permite el control de un display con dígitos de 7 segmentos agrupados y controlados con un integrado de barrido. Para su funcionamiento, utiliza el systick y escribe de un led a la vez a altas velocidades. La velocidad de escritura depende de la frecuencia del systick y del valor asignado a m_ticks. Para ver mejores resultados modificar dicho valor.

6.10.2 Constructor & Destructor Documentation

6.10.2.1 display7Segmentos()

```
display7Segmentos::display7Segmentos (
            vector< gruposdedigitos * > g,
            segmentos * s,
            barrido * b,
            const uint8_t * PosicionRelativa,
            const digito::codigo_t sistema)
```

Objeto del tipo display7Segmentos.

Crea el objeto display7Segmentos con los parámetros indicados.

Parameters

in	*g*	Vector con los dígitos a utilizar.
in	*s*	Segmento a utilizar.
in	*b*	Dispositivo de barrido a utilizar.
in	*PosicionRelativa*	Posición relativa del grupo de dígitos.
in	*sistema*	Sistema a utilizar.

6.10.2.2 ~display7Segmentos()

```
display7Segmentos::~display7Segmentos ()  [virtual]
```

destructor por defecto

6.10.3 Member Function Documentation

6.10.3.1 Clear()

```
void display7Segmentos::Clear (
            void )  [virtual]
```

Función de limpieza del display de 7 segmentos.

Coloca en 0 todos los displays.

Implements Display.

6.10.3.2 Set()

```
void display7Segmentos::Set (
            uint32_t valor,
            uint8_t dsp)
```

Función de escritura del display de 7 segmentos.

Parameters

in	*valor*	valor a escribir.
in	*dsp*	posición a escibir el valor.

6.10.3.3 SWhandler()

```
void display7Segmentos::SWhandler (
            void )  [virtual]
```

Handler del display de 7 segmentos.

Funcion interrupción del display de 7 segmentos.

Implements Callback.

6.10.3.4 Write()

```
void display7Segmentos::Write (
            const uint32_t valor)  [virtual]
```

Función de escritura del display de 7 segmentos.

Escribe el display 7 segmentos desde la posicion 0.

Parameters

in	*valor*	valor a escribir.

Implements Display.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/C-↩ Display/Display7Segmentos.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/C-↩ Display/Display7Segmentos.cpp

6.11 distancia Class Reference

Clase del objeto distancia Clase abstracta pura para la generación de HCS-R04.

```
#include <distancia.h>
```

Inheritance diagram for distancia:

Public Member Functions

- distancia ()=default
- virtual uint32_t GetDistancia ()=0
- virtual bool operator== (uint32_t a)=0
- virtual ~distancia ()=default

6.11.1 Detailed Description

Clase del objeto distancia Clase abstracta pura para la generación de HCS-R04.

6.11.2 Constructor & Destructor Documentation

6.11.2.1 distancia()

```
distancia::distancia ()  [default]
```

Constructor por defecto

6.11.2.2 ~distancia()

```
virtual distancia::~distancia ()  [virtual], [default]
```

Destructor por defecto

6.11.3 Member Function Documentation

6.11.3.1 GetDistancia()

```
virtual uint32_t distancia::GetDistancia ()  [pure virtual]
```

Obtencion de distancia

Implemented in HC_SR04.

6.11.3.2 operator==()

```
virtual bool distancia::operator== (
            uint32_t a)  [pure virtual]
```

Consulta sobre la distancia

Implemented in HC_SR04.

The documentation for this class was generated from the following file:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/07-hc-sr04/distancia.h

6.12 ESP8266 Class Reference

Clase del objeto ESP8266 El objeto ESP8266 permite la simple utilización del módulo arduino ESP8266 y el ESP01 mediante comandos AT. El módulo debe estar por defecto en la velocidad DEFAULT_ESP01_BAUDRATE. El módulo será conectado como cliente en modo TCP/UDP y con transmisión libre, sin filtros. La data llega y se envía cruda (como está). Por falta de material la clase no fue probada por completo. Sí se probó la inicialización y conección a internet, no se probó la conección a un servidor. Todas sus funciones son bloqueantes o poseen un timeout, debe ser tenido en cuenta a la hora de utilizar este driver.

`#include <ESP8266.h>`

Inheritance diagram for ESP8266:

Collaboration diagram for ESP8266:

Public Types

- enum conection_type { **TCP** = 0 , **UDP** = 1 }
- enum status_type {
  ERROR = 0 , **NOTHING** , **INITIALIZED** , **CONNECT_TO_WIFI** ,
  CONNECT_TO_SERVER }

Public Member Functions

- **ESP8266** (Pin::port_t _portTx, uint8_t _pinTx, Pin::port_t _portRx, uint8_t _pinRx, USART_Type *usart, uint32_t baudrate)

  Default constructor.
- void Inicializar (void)

  Inicializa el ESP8266/ ESP01 con comandos AT en modo cliente.
- status_type ConnectToWifi (const int8_t *wifi_address, const int8_t *wifi_password, uint32_t seg_↩ timeout=SEG_ESP01_TIMEOUT)

  Se conecta al wifi indicado.
- void DisconnectToWifi (void)

  Se desconecta de la red wifi.
- void SetIP (int8_t *ip)

  Setea la IP del dispositivo.
- int8_t * GetIP (void) const

  Devuelve la IP del dispositivo.
- bool ConnectToServer (conection_type _mode, const int8_t *server_ip, const int8_t *server_port, uint32_t seg_timeout=SEG_ESP01_TIMEOUT)

  Intenta conectarse a un servidor.
- void DisconnectToServer (void)

  Se desconecta del server.
- void Transmit (const char *msg)

  Sobrecarga de Transmit de UART.
- void Transmit (const void *msg, uint32_t n)

  Sobrecarga de Transmit de UART.
- void * Message (void *msg, uint32_t n)

  Sobrecarga de message de UART.
- status_type GetStatus (void) const

  Devuelve el estado del dispositivo.
- bool IsConnectedToWifi (void) const

  Pregunta si está conectado a internet o no.
- bool IsConnectedToServer (void) const

  Pregunta si está conectado al server o no.
- virtual ~**ESP8266** ()

  Default destructor.

Additional Inherited Members

Protected Types inherited from Uart

- enum paridad_t { **NoParidad** , **par** = 2 , **impar** }
- enum bits_de_datos { **siete_bits** , **ocho_bits** }

Protected Member Functions inherited from Uart

- Uart (Pin::port_t portTx, uint8_t pinTx, Pin::port_t portRx, uint8_t pinRx, USART_Type *usart, uint32_t baudrate, bits_de_datos BitsDeDatos, paridad_t paridad, uint32_t maxRx, uint32_t maxTx)

  Constructor de la clase uart.
- void Transmit (const char *msg) override

  Transmite el mensaje indicado.
- void Transmit (const void *msg, uint32_t n) override

  Transmite el mensaje indicado.
- void * Message (void *msg, uint32_t n) override

  Devuelve el mensaje recibido.
- void SetBaudRate (uint32_t baudrate)

  Modifica el baudrate al vuelo.

- ComunicacionAsincronica ()=default
- virtual ~ComunicacionAsincronica ()=default

6.12.1 Detailed Description

Clase del objeto ESP8266 El objeto ESP8266 permite la simple utilización del módulo arduino ESP8266 y el ESP01 mediante comandos AT. El módulo debe estar por defecto en la velocidad DEFAULT_ESP01_BAUDRATE. El módulo será conectado como cliente en modo TCP/UDP y con transmisión libre, sin filtros. La data llega y se envía cruda (como está). Por falta de material la clase no fue probada por completo. Sí se probó la inicialización y conección a internet, no se probó la conección a un servidor. Todas sus funciones son bloqueantes o poseen un timeout, debe ser tenido en cuenta a la hora de utilizar este driver.

6.12.2 Member Enumeration Documentation

6.12.2.1 conection_type

`enum ESP8266::conection_type`

Tipo de conexión del cliente wifi

6.12.2.2 status_type

`enum ESP8266::status_type`

Enumeración con todos los estados posibles de la comunicación

6.12.3 Member Function Documentation

6.12.3.1 ConnectToServer()

```
bool ESP8266::ConnectToServer (
            conection_type _mode,
            const int8_t * server_ip,
            const int8_t * server_port,
            uint32_t seg_timeout = SEG_ESP01_TIMEOUT)
```

Intenta conectarse a un servidor.

Envía los comandos AT para conectarse a un servidor. Función bloqueante, tiene un time out en caso de fallar.

Parameters

in	*_mode*	Tipo de conección UDP/TCP.
in	*server_ip*	IP/nombre del servidor.
in	*server_port*	Puerto al que conectarse del servidor.
in	*seg_timeout*	Tiempo del timeout. Default de 10 segundos.

Returns

bool: true si logró conectarse; false si no

6.12.3.2 ConnectToWifi()

```
ESP8266::status_type ESP8266::ConnectToWifi (
            const int8_t * wifi_address,
            const int8_t * wifi_password,
            uint32_t seg_timeout = SEG_ESP01_TIMEOUT)
```

Se conecta al wifi indicado.

Explicacion detallada

Parameters

in	*wifi_address*	Nombre de la red wifi.
in	*wifi_password*	Nombre de la clave de la red wifi.
in	*seg_timeout*	Tiempo hasta considerar fallo en la conexion.

Returns

status_type = CONNECT_TO_WIFI (3) si funciono, ERROR (0) sino

6.12.3.3 DisconnectToServer()

```
void ESP8266::DisconnectToServer (
            void )
```

Se desconecta del server.

Envía los comandos AT para desconectarse del servidor en caso de estar conectado a uno.

6.12.3.4 DisconnectToWifi()

```
void ESP8266::DisconnectToWifi (
            void )
```

Se desconecta de la red wifi.

Envía los comandos AT para desconectarse de la red wifi en caso de estar conectada a una.

6.12.3.5 GetIP()

```
int8_t * ESP8266::GetIP (
            void ) const
```

Devuelve la IP del dispositivo.

Returns

int8_t*: IP del dispositivo en formato string

6.12.3.6 GetStatus()

```
ESP8266::status_type ESP8266::GetStatus (
            void ) const
```

Devuelve el estado del dispositivo.

Returns

ESP8266::status_type: Estado del dispositivo.

6.12.3.7 Inicializar()

```
void ESP8266::Inicializar (
            void )
```

Inicializa el ESP8266/ ESP01 con comandos AT en modo cliente.

Configura los baudios y modo del dispositivo. FUNCION BLOQUEANTE

6.12.3.8 IsConnectedToServer()

```
bool ESP8266::IsConnectedToServer (
            void ) const
```

Pregunta si está conectado al server o no.

Returns

bool: true está conectado al server, false no lo esta.

6.12.3.9 IsConnectedToWifi()

```
bool ESP8266::IsConnectedToWifi (
            void ) const
```

Pregunta si está conectado a internet o no.

Returns

bool: true está conectado a internet, false no lo esta.

6.12.3.10 Message()

```
void * ESP8266::Message (
            void * msg,
            uint32_t n)  [virtual]
```

Sobrecarga de message de UART.

Parameters

in	*msg*	Mensaje a leer.
in	*n*	cantidad de caracteres a leer.

Returns

void∗: puntero al mensaje a leer.

Implements ComunicacionAsincronica.

6.12.3.11 SetIP()

```
void ESP8266::SetIP (
            int8_t * ip)
```

Setea la IP del dispositivo.

Envía todos los comandos AT para setear una IP y espera su correcta respuesta

Parameters

in	*ip*	ip a configurar

6.12.3.12 Transmit() [1/2]

```
void ESP8266::Transmit (
            const char * msg)  [virtual]
```

Sobrecarga de Transmit de UART.

Parameters

in	*msg*	Mensaje a enviar.

Implements ComunicacionAsincronica.

6.12.3.13 Transmit() [2/2]

```
void ESP8266::Transmit (
            const void * msg,
            uint32_t n)  [virtual]
```

Sobrecarga de Transmit de UART.

Parameters

in	*msg*	Mensaje a enviar.
in	*n*	cantidad de caracteres a enviar.

Implements ComunicacionAsincronica.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/08-WiFi/ESP8266.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/08-WiFi/ESP8266.cpp

6.13 gpio Class Reference

Clase del objeto gpio.

```
#include <gpio.h>
```

Inheritance diagram for gpio:

Collaboration diagram for gpio:

Public Types

- enum direction_t { **input** , **output** }

  Enumeracion de input/output.
- enum power_t { **off** , **on** }

  Enumeracion de encendido/apagado.
- enum mode_t {
  pushpull = 0 , **opencolector** , **inactive** = 0 , **pulldown** ,
  pullup , **repeater** }

  Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- enum activity_t { **low** , **high** }

  Enumeracion de activo bajo/alto.
- enum interrupt_mode_t {
  rising_edge = 0 , **falling_edge** , **rising_falling_edge** , **low_level** ,
  high_level }

  Enumeracion de interrupciones del pin.
- typedef enum gpio::direction_t **direction_t**

  Enumeracion de input/output.
- typedef enum gpio::mode_t **mode_t**

  Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- typedef enum gpio::activity_t **activity_t**

  Enumeracion de activo bajo/alto.

Public Types inherited from Pin

- enum port_t { **port0** , **port1** }
- enum max_bits_port_t { **b_port0** = 31 , **b_port1** = 9 }

  máximos pines por puerto
- enum error_t { **error** = 2 , **ok** }
- typedef enum Pin::port_t port_t
- typedef enum Pin::error_t error_t

Public Member Functions

- gpio (port_t port, uint8_t bit, mode_t mode, direction_t direction, activity_t activity=high)

  Constructor de clase GPIO.
- uint8_t SetPin (void) override

  Enciende la salida.
- uint8_t ClrPin (void) override

  Limpia el pin.
- uint8_t SetTogglePin (void) override

  Toggle del pin.
- uint8_t SetDir (void) override

  Setea la direccion.
- uint8_t SetToggleDir (void) override

  Toggle de la dirección.
- uint8_t GetPin (void) override

  Devuelve el valor del pin.
- uint8_t SetPinMode (void) override

  Configura el modo del pin.
- uint8_t SetPinResistor (void) override

  Setea la resistencia interna.
- gpio & operator= (uint8_t a)

  Sobrecarga del operador =.
- virtual ~gpio ()=default

Public Member Functions inherited from InOut

- InOut ()=default
- virtual ~InOut ()=default

Public Member Functions inherited from Pin

- Pin (port_t port, uint8_t bit)

  Constructor de clase PIN.

Protected Attributes

- const mode_t m_mode
- direction_t m_direction
- const activity_t m_activity

Additional Inherited Members

Public Attributes inherited from Pin

- const port_t m_port
- const uint8_t m_bit
- int8_t m_error

6.13.1 Detailed Description

Clase del objeto gpio.

El objeto gpio Permite el manejo de entradas y salidas de uso general

6.13.2 Constructor & Destructor Documentation

6.13.2.1 gpio()

```
gpio::gpio (
            port_t port,
            uint8_t bit,
            mode_t mode,
            direction_t direction,
            activity_t activity = high)
```

Constructor de clase GPIO.

Crea un GPIO con los parámetros correspondientes

Parameters

in	*port*	Puerto del objeto
in	*bit*	Bit del objeto
in	*mode*	Configuración eléctrica del pin
in	*direction*	Entrada/salida
in	*activity*	Activo alto/bajo

6.13.2.2 ~gpio()

```
virtual gpio::~gpio ()  [virtual], [default]
```

Destructor por defecto

6.13.3 Member Function Documentation

6.13.3.1 ClrPin()

```
uint8_t gpio::ClrPin (
            void )  [override], [virtual]
```

Limpia el pin.

Coloca la salida con un 0 lógico

Returns

Devuelve el error

Implements InOut.

6.13.3.2 GetPin()

```
uint8_t gpio::GetPin (
            void )  [override], [virtual]
```

Devuelve el valor del pin.

Entrega el valor lógico de encendido o apagado de la entrada

Returns

Devuelve el error

Implements InOut.

6.13.3.3 operator=()

```
gpio & gpio::operator= (
            uint8_t a)
```

Sobrecarga del operador =.

Enciende o apaga la salida. Solo funciona en modo output

Parameters

in	*a*	Valor de seteo de la salida

Returns

Referencia a si mismo

6.13.3.4 SetDir()

```
uint8_t gpio::SetDir (
            void )  [override], [virtual]
```

Setea la direccion.

Configura la GPIO como entrada o como salida

Returns

Devuelve el error

Implements InOut.

6.13.3.5 SetPin()

```
uint8_t gpio::SetPin (
            void )  [override], [virtual]
```

Enciende la salida.

Coloca la salida con un 1 lógico

Returns

Devuelve el error

Implements InOut.

6.13.3.6 SetPinMode()

```
uint8_t gpio::SetPinMode (
            void )  [override], [virtual]
```

Configura el modo del pin.

Activa el pin en modo PushPull o OpenCollector. SOLO PARA OUTPUTS

Returns

Devuelve el error

Implements InOut.

6.13.3.7 SetPinResistor()

```
uint8_t gpio::SetPinResistor (
            void )  [override], [virtual]
```

Setea la resistencia interna.

Activa el modo Inactive , PullUp , PullDown o Repeater. SOLO PARA INPUTS

Returns

Devuelve el error

Implements InOut.

6.13.3.8 SetToggleDir()

```
uint8_t gpio::SetToggleDir (
            void )  [override], [virtual]
```

Toggle de la dirección.

Pasa de output a input y viceversa

Returns

Devuelve el error

Implements InOut.

6.13.3.9 SetTogglePin()

```
uint8_t gpio::SetTogglePin (
            void )  [override], [virtual]
```

Toggle del pin.

En caso de ser salida. Pasa de encendido a apagado y viceversa.

Returns

Devuelve el error

Implements InOut.

6.13.4 Member Data Documentation

6.13.4.1 m_activity

```
const activity_t gpio::m_activity  [protected]
```

Si es activo alto o activo bajo

6.13.4.2 m_direction

```
direction_t gpio::m_direction  [protected]
```

Si es input u output

6.13.4.3 m_mode

```
const mode_t gpio::m_mode  [protected]
```

La config de resistencia y actividad

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/02-Gpio/gpio.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/02-Gpio/gpio.cpp

6.14 gruposdedigitos Struct Reference

Estructura de grupo de dígitos.

```
#include <GrupoDeDigitos.h>
```

Public Member Functions

- gruposdedigitos (uint8_t comienzo, uint8_t cantidad)

Public Attributes

- const uint8_t m_comienzo
- const uint8_t m_cantidad

6.14.1 Detailed Description

Estructura de grupo de dígitos.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 gruposdedigitos()

```
gruposdedigitos::gruposdedigitos (
            uint8_t comienzo,
            uint8_t cantidad)  [inline]
```

< Constructor por defecto

6.14.3 Member Data Documentation

6.14.3.1 m_cantidad

```
const uint8_t gruposdedigitos::m_cantidad
```

Cantidad de dígitos

6.14.3.2 m_comienzo

```
const uint8_t gruposdedigitos::m_comienzo
```

Comienzo del grupo

The documentation for this struct was generated from the following file:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/C-↩ Display/GrupoDeDigitos.h

6.15 HC_SR04 Class Reference

Clase del objeto HC_SR04 El objeto HC_SR04 Mide distancia mediante el uso de un ultrasónico. Debido a los tiempos muy pequeños de uso, no se recomienda utilizar en grandes cantidades.

```
#include <HCSR04.h>
```

Inheritance diagram for HC_SR04:

Collaboration diagram for HC_SR04:

Public Member Functions

- HC_SR04 (PWM_Reader *&rx, Pwm *&tx)
  Constructor de clase HC_SR04.
- void Inicializar (void)
  Inicializa todas las patas del ultrasónico.
- uint32_t GetDistancia (void)
  Devuelve la distancia.
- void Off (void)
  Apago las patas del ultrasonico.
- void On (void)
  Enciendo las patas del ultrasonico.
- bool operator== (const uint32_t a)
  Operador ==.
- bool operator<= (const uint32_t a)
  Verifica si la distancia es menor o igual que a.
- bool operator>= (const uint32_t a)
  Verifica si la distancia es mayor o igual que a.
- bool operator< (const uint32_t a)
  Verifica si la distancia es menor que a.
- bool operator> (const uint32_t a)
  Verifica si la distancia es mayor que a.

Additional Inherited Members

Protected Member Functions inherited from distancia

- distancia ()=default
- virtual ~distancia ()=default

6.15.1 Detailed Description

Clase del objeto HC_SR04 El objeto HC_SR04 Mide distancia mediante el uso de un ultrasónico. Debido a los tiempos muy pequeños de uso, no se recomienda utilizar en grandes cantidades.

SENSOR ULTRASÓNICO HC-SR04: Se le debe enviar PWM de 60ms o más (recomendamos 80) con un tiempo de encendido de 10us. Recebirá otro pulso variable que debe ser leído por tx. Devuelve el valor de distancia en cm (centímetros)

6.15.2 Constructor & Destructor Documentation

6.15.2.1 HC_SR04()

```
HC_SR04::HC_SR04 (
            PWM_Reader *& rx,
            Pwm *& tx)
```

Constructor de clase HC_SR04.

Crea un HC_SR04 con los parámetros correspondientes

Parameters

in	*rx*	Lector de anchos de pulso utilizado en el pin ECHO.
in	*tx*	PWM utilizado como pin TRIGG.

6.15.3 Member Function Documentation

6.15.3.1 GetDistancia()

```
uint32_t HC_SR04::GetDistancia (
            void )  [virtual]
```

Devuelve la distancia.

A menos que esté apagado, devuelve el valor de distancia obtenido. Para evitar overflow del Uint32, se verifica el valor censado.

Returns

m_distancia

Implements distancia.

6.15.3.2 Inicializar()

```
void HC_SR04::Inicializar (
            void )
```

Inicializa todas las patas del ultrasónico.

Inicializa el PWM con el valor correspondiente y el contador de ancho ancho de pulso.

6.15.3.3 Off()

```
void HC_SR04::Off (
            void )
```

Apago las patas del ultrasonico.

Llamo a las funciones de apagar y guardo en el buffer un valor imposible.

6.15.3.4 On()

```
void HC_SR04::On (
            void )
```

Enciendo las patas del ultrasonico.

Llamo a las funciones de encender de tx y rx.

6.15.3.5 operator<()

```
bool HC_SR04::operator< (
            const uint32_t a)
```

Verifica si la distancia es menor que a.

Realiza el cálculo de distancia actual y luego lo compara con a.

Parameters

in	*a*	Distancia a comparar.

Returns

bool

6.15.3.6 operator<=()

```
bool HC_SR04::operator<= (
            const uint32_t a)
```

Verifica si la distancia es menor o igual que a.

Realiza el cálculo de distancia actual y luego lo compara con a.

Parameters

in	*a*	Distancia a comparar.

Returns

bool

6.15.3.7 operator==()

```
bool HC_SR04::operator== (
            const uint32_t a)  [virtual]
```

Operador ==.

Consulta si la distancia es igual o no a la entregada.

Parameters

in	*a*	Valor de distancia.

Returns

bool

Implements distancia.

6.15.3.8 operator>()

```
bool HC_SR04::operator> (
            const uint32_t a)
```

Verifica si la distancia es mayor que a.

Realiza el cálculo de distancia actual y luego lo compara con a.

Parameters

in	*a*	Distancia a comparar.

Returns

bool

6.15.3.9 operator>=()

```
bool HC_SR04::operator>= (
            const uint32_t a)
```

Verifica si la distancia es mayor o igual que a.

Realiza el cálculo de distancia actual y luego lo compara con a.

Parameters

in	*a*	Distancia a comparar.

Returns

bool

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/07-hc-sr04/HCSR04.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/07-hc-sr04/HCSR04.cpp

6.16 I2C Class Reference

Clase del objeto I2C El objeto I2C genera una comunicación sincrónica de tipo I2C. Posee las funciones basicas como start, stop, write y read.

`#include <I2C.h>`

Inheritance diagram for I2C:

Collaboration diagram for I2C:

Public Types

- enum I2C_mode_t { **master** = 1 , **slave** = 2 }

  Enumeracion. Modo del I2C. Puede ser master o slave.
- enum I2C_action_t { **write** = 0 , **read** = 1 }

  Enumeracion. Acciones del I2C. Pueden ser leer o escribir (write o read).
- enum I2C_states_t {
  idle = 0 , **rx_data** = 1 , **tx_ready** , **NACK_addr** ,
  NACK_tx , **slvst_addr** = 0 , **slvst_rx** , **slvst_tx** ,
  busy = 10 }

  Enumeracion. Estados del I2C.

Public Member Functions

- I2C (I2C_Type *I2C_register, Pin *sda, Pin *scl, I2C_mode_t mode=master, uint8_t slv_addr=0)

  Constructor de la clase I2C.
- void Initialize (uint32_t clk_freq)

  Inicializa el I2C.
- void **EnableInterupt** (void)

  Habilita la interrupcion.
- void **DisableInterupt** (void)

  Deshabilita la interrupcion.
- void Start (uint8_t addr, I2C_action_t action)

  Inicia una comunicacion I2C.
- void Stop (void)

Detiene la comunicacion.

- void Write (uint8_t data) override

  Escribe un valor a un slave.
- I2C & operator= (uint8_t data)

  Sobrecarga del operador = (igual).
- int8_t Read (uint8_t *data, bool continue_reading)

  Lee un valor recibido.
- void ACK (bool a)

  Envia un Acknowledge.
- bool ACKaddr (void)

  Acknowledge especial del address.
- I2C_states_t GetState (void)

  Regresa el estado del I2C.
- virtual void I2C_IRQHandler (void)

6.16.1 Detailed Description

Clase del objeto I2C El objeto I2C genera una comunicación sincrónica de tipo I2C. Posee las funciones basicas como start, stop, write y read.

6.16.2 Member Enumeration Documentation

6.16.2.1 I2C_states_t

```
enum I2C::I2C_states_t
```

Enumeracion. Estados del I2C.

Un master puede encontrarse listo para transmitir o recibir (rx_data y tx_ready) o recibiendo un NACK (NACK_addr y NCK_tx). Un slave puede encontrarse listo para transmitir o recibir (slvst_tx y slvst_rx) o recibiendo un addres para iniciar (slvst_addr) Todos los modos pueden estar en idle si no hacen nada o busy si estan en mitad de un procesamiento.

6.16.3 Constructor & Destructor Documentation

6.16.3.1 I2C()

```
I2C::I2C (
            I2C_Type * I2C_register,
            Pin * sda,
            Pin * scl,
            I2C_mode_t mode = master,
            uint8_t slv_addr = 0)
```

Constructor de la clase I2C.

Genera un I2C con los parametros indicados

Parameters

in	*I2C_register*	Registro del I2C a utilizar. Puede valer I2C0, I2C1, I2C2 o I2C3.
in	*sda*	Pin de transmision de datos.
in	*scl*	Pin de clock.
in	*mode*	Tipo de I2C. Puede ser master o slave.
in	*slv_addr*	Address propio del I2C. Solo util cuando se lo configura como slave.

6.16.4 Member Function Documentation

6.16.4.1 ACK()

```
void I2C::ACK (
            bool a)
```

Envia un Acknowledge.

Para un I2C en modo slave, envia un Acknowledge que permite continuar la comunicacion.

Parameters

in	*a*	bool que indica si la recepcion es exitosa o no. Si es true se envia un ACK. Si es false se envia NACK.

6.16.4.2 ACKaddr()

```
bool I2C::ACKaddr (
            void )
```

Acknowledge especial del address.

Envia un acknowledge solo si el address que se recibio coindide con el propio.

6.16.4.3 GetState()

```
I2C::I2C_states_t I2C::GetState (
            void )
```

Regresa el estado del I2C.

Regresa el estado actual del I2C, ya sea master o slave. Un master puede estar rx_data si se recibio data, tx_↩ ready si esta listo para transmitir, NACK_addr o NACK_tx si recibio un NACK. Un slave puede estar en slvst_addr si recibio un start, slvst_tx si recibio un pedido a transmitir o slvst_rx si recibio un pedido a leer. Ambos pueden estar en idle si no se encuentran haciendo nada o busy si estan en mitad de un envio de informacion.

6.16.4.4 I2C_IRQHandler()

```
virtual void I2C::I2C_IRQHandler (
            void )  [inline], [virtual]
```

Handler generico de interrupcion I2C. No hace nada, debe heredarse y sobreescribirse.

6.16.4.5 Initialize()

```
void I2C::Initialize (
            uint32_t clk_freq)
```

Inicializa el I2C.

Configura todos los registros para el uso del I2C. Configura los clocks, resetea, configura la SWM, configura los registros y las interrupciones.

Parameters

in	*clk_freq*	frecuencia del clock de transmision a utilizar. En KHz

6.16.4.6 operator=()

```
I2C & I2C::operator= (
            uint8_t data)
```

Sobrecarga del operador = (igual).

Sobrecarga de operador para escritura de un byte.

Parameters

data	byte a escribir.

Returns

I2C&: Puntero a si mismo.

6.16.4.7 Read()

```
int8_t I2C::Read (
            uint8_t * data,
            bool continue_reading)
```

Lee un valor recibido.

Lee el valor recibido por un slave. Solo puede ser llamado por un master que previamente haya pedido recibir y o un slave al que le llego un valor.

Parameters

in	*data*	puntero a char donde se guardara el valor recibido.
in	*continue_reading*	bool que indica si se va a seguir leyendo o no. Si es true, se envia un "continue" y el slave volvera a transmitir informacion. Si es false no se realiza ninguna accion. Si el I2C es slave, continue_reading no tiene proposito.

Returns

int8_t: valor de error. 0 no hubo problemas. -1 no se puede leer aun.

6.16.4.8 Start()

```
void I2C::Start (
            uint8_t addr,
            I2C_action_t action)
```

Inicia una comunicacion I2C.

Envia el start a un slave. La funcion solo puede ser llamada por un master si este no se encuentra ocupado.

Parameters

in	*addr*	address del slave al que se comunicara.
in	*action*	accion a realizar. Puede ser read o write.

6.16.4.9 Stop()

```
void I2C::Stop (
            void )
```

Detiene la comunicacion.

Envia una señal de stop para detener la comunicacion. Solo puede ser llamada por un master que no se encuentre ocupado.

6.16.4.10 Write()

```
void I2C::Write (
            uint8_t data)  [override], [virtual]
```

Escribe un valor a un slave.

Envia un dato. Solo puede ser llamada por un master que previamente haya pedido transmitir informacion o un slave que este en modo recibir transmision.

Parameters

in	*data*	byte a enviar.

Implements ComunicacionSincronica.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/14-ComSincronica/I2C/I2C.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/14-ComSincronica/I2C/I2C.cpp

6.17 I2CMaster Class Reference

Clase del objeto I2CMaster El objeto I2CMaster genera una comunicación tipo master de I2C utilizando buffers de recepcion y transmision con interrupciones.

`#include <I2CMaster.h>`

Inheritance diagram for I2CMaster:

Collaboration diagram for I2CMaster:

Public Member Functions

- I2CMaster (I2C_Type *I2C_register, Pin *sda, Pin *scl, uint32_t maxRx=15, uint32_t maxTx=15)

  Constructor de la clase I2CMaster.
- void Initialize (uint32_t clk_freq)

  Inicializa el I2C.
- void Write (uint8_t addr, const char *msg)

  Transmite el mensaje indicado.
- void Write (uint8_t addr, const void *msg, uint32_t n)

  Transmite length cantidad de bytes del mensaje indicado.
- void RequestRead (uint8_t addr, uint32_t cant_read=1)

  Inicia una comunicacion de lectura.
- void * Read (void *msg, uint32_t n)

  Lee el buffer de recepcion.
- bool isIdle (void)

  Indica si el I2C esta en reposo o no.

Additional Inherited Members

Protected Types inherited from I2C

- enum I2C_mode_t { **master** = 1 , **slave** = 2 }

  Enumeracion. Modo del I2C. Puede ser master o slave.
- enum I2C_action_t { **write** = 0 , **read** = 1 }

  Enumeracion. Acciones del I2C. Pueden ser leer o escribir (write o read).
- enum I2C_states_t {
  idle = 0 , **rx_data** = 1 , **tx_ready** , **NACK_addr** ,
  NACK_tx , **slvst_addr** = 0 , **slvst_rx** , **slvst_tx** ,
  busy = 10 }

  Enumeracion. Estados del I2C.

Protected Member Functions inherited from I2C

- I2C (I2C_Type *I2C_register, Pin *sda, Pin *scl, I2C_mode_t mode=master, uint8_t slv_addr=0)

  Constructor de la clase I2C.
- void Initialize (uint32_t clk_freq)

  Inicializa el I2C.
- void **EnableInterupt** (void)

  Habilita la interrupcion.
- void **DisableInterupt** (void)

  Deshabilita la interrupcion.
- void Start (uint8_t addr, I2C_action_t action)

  Inicia una comunicacion I2C.
- void Stop (void)

  Detiene la comunicacion.
- void Write (uint8_t data) override

  Escribe un valor a un slave.
- I2C & operator= (uint8_t data)

  Sobrecarga del operador = (igual).
- int8_t Read (uint8_t *data, bool continue_reading)

  Lee un valor recibido.
- void ACK (bool a)

  Envia un Acknowledge.
- bool ACKaddr (void)

  Acknowledge especial del address.
- I2C_states_t GetState (void)

  Regresa el estado del I2C.

6.17.1 Detailed Description

Clase del objeto I2CMaster El objeto I2CMaster genera una comunicación tipo master de I2C utilizando buffers de recepcion y transmision con interrupciones.

6.17.2 Constructor & Destructor Documentation

6.17.2.1 I2CMaster()

```
I2CMaster::I2CMaster (
            I2C_Type * I2C_register,
            Pin * sda,
            Pin * scl,
            uint32_t maxRx = 15,
            uint32_t maxTx = 15)
```

Constructor de la clase I2CMaster.

Genera un master I2C con buffers de transmision y recepcion para envio de strings.

Parameters

in	*I2C_register*	Registro del I2C a utilizar. Puede valer I2C0, I2C1, I2C2 o I2C3.
in	*sda*	Pin de transmision de datos.
in	*scl*	Pin de clock.
in	*maxRx*	Valor maximo del registro de recepcion. Por default 15.
in	*maxTx*	Valor maximo del registro de transmision. Por default 15.

6.17.3 Member Function Documentation

6.17.3.1 Initialize()

```
void I2CMaster::Initialize (
            uint32_t clk_freq)
```

Inicializa el I2C.

Configura todos los registros para el uso del I2C. Configura los clocks, resetea, configura la SWM, configura los registros y las interrupciones.

Parameters

clk_freq	frecuencia del clock de transmision a utilizar.

6.17.3.2 isIdle()

```
bool I2CMaster::isIdle (
            void )
```

Indica si el I2C esta en reposo o no.

Indica si el estado del I2C es idle. Solo se puede enviar y recibir informacion si se encuentra en idle.

Returns

bool: true si esta en idle, false si no lo esta.

6.17.3.3 Read()

```
void * I2CMaster::Read (
            void * msg,
            uint32_t length)
```

Lee el buffer de recepcion.

Lee una cantidad fija de bytes dentro del buffer de recepcion.

Parameters

in	*msg*	Puntero a variable donde se enviaran los valores leidos.
in	*length*	Cantidad de bytes a leer.

Returns

void∗: Puntero a los valores leidos. Si hubo error se envia nullptr.

6.17.3.4 RequestRead()

```
void I2CMaster::RequestRead (
            uint8_t addr,
            uint32_t cant_read = 1)
```

Inicia una comunicacion de lectura.

Comienza una comunicacion en modo lectura que no se detiene hasta recibir cant_read bytes. Los bytes leidos se guardaran en el buffer.

Parameters

in	*addr*	address del slave al que se comunicara.
in	*cant_read*	cantidad de bytes a transmitir.

6.17.3.5 Write() [1/2]

```
void I2CMaster::Write (
            uint8_t addr,
            const char * msg)
```

Transmite el mensaje indicado.

Coloca el mensaje indicado en el buffer. Importante, el mensaje debe terminar en \0 (String).

Parameters

in	*addr*	address del slave al que se comunicara.
in	*msg*	Mensaje a transmitir.

6.17.3.6 Write() [2/2]

```
void I2CMaster::Write (
            uint8_t addr,
            const void * msg,
            uint32_t length)
```

Transmite length cantidad de bytes del mensaje indicado.

Coloca una cantidad fija del mensaje indicado en el buffer.

Parameters

in	*addr*	address del slave al que se comunicara.
in	*msg*	Mensaje a transmitir.
in	*length*	cantidad de bytes a transmitir.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/14-ComSincronica/I2C/I2CMaster.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/14-ComSincronica/I2C/I2↩ CMaster.cpp

6.18 I4017 Class Reference

Clase del objeto I4017 El objeto I4017 permite el control del integrado del mismo nombre. Habitualmente utilizado para barrer información a tavés de sus patas.

```
#include <I4017.h>
```

Inheritance diagram for I4017:

Collaboration diagram for I4017:

Public Member Functions

- I4017 (const vector< gpio * > &pins4017, uint8_t maxsalidas)
- void SetDigito (void)

  Función de cambio en el barrido.
- void SetReset (void)

  Resetea el I4017.
- void SetClock (void)

  Enciende la pata clock.
- void Inicializar (void)

  Función de Inicialización del I4017.

6.18.1 Detailed Description

Clase del objeto I4017 El objeto I4017 permite el control del integrado del mismo nombre. Habitualmente utilizado para barrer información a tavés de sus patas.

6.18.2 Constructor & Destructor Documentation

6.18.2.1 I4017()

```
I4017::I4017 (
            const vector< gpio * > & pins4017,
            uint8_t maxsalidas)  [inline]
```

constructor por defecto

6.18.3 Member Function Documentation

6.18.3.1 Inicializar()

```
void I4017::Inicializar (
            void )  [virtual]
```

Función de Inicialización del I4017.

Implements barrido.

6.18.3.2 SetClock()

```
void I4017::SetClock (
            void )
```

Enciende la pata clock.

El I4017 aumenta en 1 la salida al recibir un pulso de clock.

6.18.3.3 SetDigito()

```
void I4017::SetDigito (
            void )  [virtual]
```

Función de cambio en el barrido.

Se mueve 1 en el barrido de bits.

Implements barrido.

6.18.3.4 SetReset()

```
void I4017::SetReset (
            void )
```

Resetea el I4017.

El I4017 vuelve al valor 0 de salida por enviar un pulso a la pata de Reset.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/A-↩ Barrido/I4017.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/A-↩ Barrido/I4017.cpp

6.19 I4511 Class Reference

Clase del objeto I4511 El objeto I4511 permite el control del integrado del mismo nombre. Este integrado permite el control de un display 7 segmentos mediante una comunicación binaria en formato paralelo.

`#include <I4511.h>`

Inheritance diagram for I4511:

Collaboration diagram for I4511:

Public Member Functions

- I4511 (const vector< gpio * > &bcd)
- void SetSegmentos (uint16_t)

  Función de escritura del segmento.
- void Inicializar (void)

  Función de Inicialización del I4511.

6.19.1 Detailed Description

Clase del objeto I4511 El objeto I4511 permite el control del integrado del mismo nombre. Este integrado permite el control de un display 7 segmentos mediante una comunicación binaria en formato paralelo.

6.19.2 Constructor & Destructor Documentation

6.19.2.1 I4511()

```
I4511::I4511 (
            const vector< gpio * > & bcd)  [inline]
```

constructor por defecto

6.19.3 Member Function Documentation

6.19.3.1 Inicializar()

```
void I4511::Inicializar (
            void )  [virtual]
```

Función de Inicialización del I4511.

Implements segmentos.

6.19.3.2 SetSegmentos()

```
void I4511::SetSegmentos (
            uint16_t valor)  [virtual]
```

Función de escritura del segmento.

Escribe el segmento en formato binario.

Parameters

in	*valor*	valor a escribir.

Implements segmentos.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/B-Segmentos/I4511.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/B-Segmentos/I4511.cpp

6.20 InOut Class Reference

Clase del objeto InOut.

`#include <InOut.h>`

Inheritance diagram for InOut:

Public Member Functions

- InOut ()=default
- virtual uint8_t SetPin (void)=0
- virtual uint8_t ClrPin (void)=0
- virtual uint8_t SetTogglePin (void)=0
- virtual uint8_t SetDir (void)=0
- virtual uint8_t SetToggleDir (void)=0
- virtual uint8_t GetPin (void)=0
- virtual uint8_t SetPinMode (void)=0
- virtual uint8_t SetPinResistor (void)=0
- virtual ~InOut ()=default

6.20.1 Detailed Description

Clase del objeto InOut.

Clase abstracta pura para la generación de GPIO

6.20.2 Constructor & Destructor Documentation

6.20.2.1 InOut()

```
InOut::InOut ()  [default]
```

Constructor por defecto

6.20.2.2 ~InOut()

```
virtual InOut::~InOut ()  [virtual], [default]
```

Destructor por defecto

6.20.3 Member Function Documentation

6.20.3.1 ClrPin()

```
virtual uint8_t InOut::ClrPin (
            void )  [pure virtual]
```

Apaga el pin

Implemented in gpio.

6.20.3.2 GetPin()

```
virtual uint8_t InOut::GetPin (
            void )  [pure virtual]
```

Obtiene valor del pin

Implemented in gpio.

6.20.3.3 SetDir()

```
virtual uint8_t InOut::SetDir (
            void )  [pure virtual]
```

Setea la direccion del pin

Implemented in gpio.

6.20.3.4 SetPin()

```
virtual uint8_t InOut::SetPin (
            void )  [pure virtual]
```

Enciende el pin

Implemented in gpio.

6.20.3.5 SetPinMode()

```
virtual uint8_t InOut::SetPinMode (
            void )  [pure virtual]
```

Setea modo del pin

Implemented in gpio.

6.20.3.6 SetPinResistor()

```
virtual uint8_t InOut::SetPinResistor (
            void )  [pure virtual]
```

Setea resistencia del pin

Implemented in gpio.

6.20.3.7 SetToggleDir()

```
virtual uint8_t InOut::SetToggleDir (
            void )  [pure virtual]
```

Cambia la direccion del pin

Implemented in gpio.

6.20.3.8 SetTogglePin()

```
virtual uint8_t InOut::SetTogglePin (
            void )  [pure virtual]
```

Cambia el valor del pin

Implemented in gpio.

The documentation for this class was generated from the following file:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/02-Gpio/InOut.h

6.21 Input Class Reference

Clase del objeto Input.

```
#include <Input.h>
```

Inheritance diagram for Input:

Collaboration diagram for Input:

Public Member Functions

- Input (port_t puerto, uint8_t bit, mode_t modo, activity_t actividad=high, uint8_t MaxBounce=MAX_BOUNCE)
  Constructor de clase Input.
- void Inicializar (void)
  Inicializa la entrada.
- uint8_t get (void)
  Devuelve el valor de la input.
- void SWhandler (void) override
  Funcion de interrupcion del systick.
- bool operator== (uint8_t val)
  Sobrecarga de del operador de igualdad.
- bool operator!= (uint8_t val)
  Sobrecarga de del operador de desigualdad.
- virtual ~Input ()

Public Member Functions inherited from Callback

- void SetInterrupt ()
  Activa la interrupción.
- void UnSetInterrupt ()
  Desactiva la interrupción.

Friends

- bool operator== (uint32_t val, Input &I)
  Sobrecarga de del operador de asignacion.

Additional Inherited Members

Protected Types inherited from gpio

- enum direction_t { **input** , **output** }

  Enumeracion de input/output.
- enum power_t { **off** , **on** }

  Enumeracion de encendido/apagado.
- enum mode_t {
  pushpull = 0 , **opencolector** , **inactive** = 0 , **pulldown** ,
  pullup , **repeater** }

  Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- enum activity_t { **low** , **high** }

  Enumeracion de activo bajo/alto.
- enum interrupt_mode_t {
  rising_edge = 0 , **falling_edge** , **rising_falling_edge** , **low_level** ,
  high_level }

  Enumeracion de interrupciones del pin.
- typedef enum gpio::direction_t **direction_t**

  Enumeracion de input/output.
- typedef enum gpio::mode_t **mode_t**

  Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- typedef enum gpio::activity_t **activity_t**

  Enumeracion de activo bajo/alto.

Protected Types inherited from Pin

- enum port_t { **port0** , **port1** }
- enum max_bits_port_t { **b_port0** = 31 , **b_port1** = 9 }

  máximos pines por puerto
- enum error_t { **error** = 2 , **ok** }
- typedef enum Pin::port_t port_t
- typedef enum Pin::error_t error_t

Protected Member Functions inherited from gpio

- gpio (port_t port, uint8_t bit, mode_t mode, direction_t direction, activity_t activity=high)

  Constructor de clase GPIO.
- uint8_t SetPin (void) override

  Enciende la salida.
- uint8_t ClrPin (void) override

  Limpia el pin.
- uint8_t SetTogglePin (void) override

  Toggle del pin.
- uint8_t SetDir (void) override

  Setea la direccion.
- uint8_t SetToggleDir (void) override

  Toggle de la dirección.
- uint8_t GetPin (void) override

  Devuelve el valor del pin.

- uint8_t SetPinMode (void) override
  Configura el modo del pin.
- uint8_t SetPinResistor (void) override
  Setea la resistencia interna.
- gpio & operator= (uint8_t a)
  Sobrecarga del operador =.
- virtual ~gpio ()=default

Protected Member Functions inherited from InOut

- InOut ()=default
- virtual ~InOut ()=default

Protected Member Functions inherited from Pin

- Pin (port_t port, uint8_t bit)
  Constructor de clase PIN.

Protected Member Functions inherited from Callback

- void SetInterrupt ()
  Activa la interrupción.
- void UnSetInterrupt ()
  Desactiva la interrupción.

Protected Attributes inherited from gpio

- const mode_t m_mode
- direction_t m_direction
- const activity_t m_activity

Protected Attributes inherited from Pin

- const port_t m_port
- const uint8_t m_bit
- int8_t m_error

6.21.1 Detailed Description

Clase del objeto Input.

El objeto Input Permite el manejo de entradas digitales con un antirrebote propio por software.

6.21.2 Constructor & Destructor Documentation

6.21.2.1 Input()

```
Input::Input (
            port_t puerto,
            uint8_t bit,
            mode_t modo,
            activity_t actividad = high,
            uint8_t MaxBounce = MAX_BOUNCE)
```

Constructor de clase Input.

Crea un Input con los parámetros correspondientes.

Parameters

in	*puerto*	Puerto del objeto.
in	*bit*	Bit del objeto.
in	*modo*	Configuracion eléctrica del pin.
in	*actividad*	Activo alto/bajo.
in	*MaxBounce*	Cantidad de rebotes permitidos.

6.21.2.2 ~Input()

```
Input::~Input ()  [virtual]
```

Destructor por defecto

6.21.3 Member Function Documentation

6.21.3.1 get()

```
uint8_t Input::get (
            void )
```

Devuelve el valor de la input.

Entrega el valor "real" de la entrada sin su rebote.

Returns

uint32_t: Valor de la entrada.

6.21.3.2 Inicializar()

```
void Input::Inicializar (
            void )
```

Inicializa la entrada.

Configura el GPIO y el buffer.

6.21.3.3 operator"!=()

```
bool Input::operator!= (
            uint8_t val)
```

Sobrecarga de del operador de desigualdad.

Parameters

in	*val*	Valor a comparar con el buffer.

Returns

true si la entrada no esta en val.

6.21.3.4 operator==()

```
bool Input::operator== (
            uint8_t val)
```

Sobrecarga de del operador de igualdad.

Parameters

in	*val*	Valor a comparar con el buffer.

Returns

true si la entrada esta en val.

6.21.3.5 SWhandler()

```
void Input::SWhandler (
            void )  [override], [virtual]
```

Funcion de interrupcion del systick.

Guarda en el buffer el valor de la entrada en caso de haber pasado el antirrebote.

Implements Callback.

6.21.4 Friends And Related Symbol Documentation

6.21.4.1 operator==

```
bool operator== (
            uint32_t val,
            Input & I)  [friend]
```

Sobrecarga de del operador de asignacion.

Parameters

in	*val*	Valor a comparar con el buffer
in	*I*	Entrada a comparar

Returns

true si la entrada esta en val

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/04-Input/Input.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/04-Input/Input.cpp

6.22 L298N Class Reference

Clase del objeto L298N El objeto L298N realiza las acciones de control de dos motores controlados por el correspondiente periférico.

`#include <L298N.h>`

Inheritance diagram for L298N:

Collaboration diagram for L298N:

Public Member Functions

- L298N (gpio *&_motorDer_a, gpio *&_motorDer_b, gpio *&_motorIzq_a, gpio *&_motorIzq_b)
  Constructor de clase L298N.
- void Inicializar (void)
  Inicializa todas las salidas GPIO.
- void GirarIzq (void)
  Gira a la izquierda.
- void GirarDer (void)
  Gira a la derecha.
- void Girar (const uint8_t direccion)
  Gira el motor.
- void Frenar (void)
  Frena el motor.
- void Avanzar (void)
  Avanza los motores.
- void Retroceder (void)
  Pone los motores marcha atrás.
- virtual ~L298N ()

Additional Inherited Members

Protected Types inherited from Puente_H

- enum { **IZQUIERDA** = 0 , **DERECHA** }

Protected Member Functions inherited from Puente_H

- Puente_H ()=default
- virtual ~Puente_H ()=default

6.22.1 Detailed Description

Clase del objeto L298N El objeto L298N realiza las acciones de control de dos motores controlados por el correspondiente periférico.

6.22.2 Constructor & Destructor Documentation

6.22.2.1 L298N()

```
L298N::L298N (
            gpio *& _motorDer_a,
            gpio *& _motorDer_b,
            gpio *& _motorIzq_a,
            gpio *& _motorIzq_b)
```

Constructor de clase L298N.

Crea un L298N con los parámetros correspondientes

Parameters

in	*_motorDer_a*	Puntero a gpio del motor derecho A.
in	*_motorDer_b*	Puntero a gpio del motor derecho B.
in	*_motorIzq_a*	Puntero a gpio del motor izquierdo A.
in	*_motorIzq_b*	Puntero a gpio del motor izquierdo B.

6.22.2.2 ~L298N()

```
L298N::~L298N ()  [virtual]
```

Destructor por defecto

6.22.3 Member Function Documentation

6.22.3.1 Avanzar()

```
void L298N::Avanzar (
            void )  [virtual]
```

Avanza los motores.

Enciende los 2 motores con tensión en la pata A.

Implements Puente_H.

6.22.3.2 Frenar()

```
void L298N::Frenar (
            void )  [virtual]
```

Frena el motor.

Deja las cuatro salidas en 0.

Implements Puente_H.

6.22.3.3 Girar()

```
void L298N::Girar (
            const uint8_t direccion)  [virtual]
```

Gira el motor.

Utilizando la enumeración de PuenteH realiza el giro correspondiente.

Parameters

in	*direccion*	Dirección del giro (IZQUIERDA = 0 , DERECHA = 1)

Implements Puente_H.

6.22.3.4 GirarDer()

```
void L298N::GirarDer (
            void )  [virtual]
```

Gira a la derecha.

Energiza solamente el motor 2 de forma directa.

Implements Puente_H.

6.22.3.5 GirarIzq()

```
void L298N::GirarIzq (
            void )  [virtual]
```

Gira a la izquierda.

Energiza solamente el motor 1 de forma directa.

Implements Puente_H.

6.22.3.6 Inicializar()

```
void L298N::Inicializar (
            void )  [virtual]
```

Inicializa todas las salidas GPIO.

Setea la dirección y apaga todas las GPIO que posee.

Implements Puente_H.

6.22.3.7 Retroceder()

```
void L298N::Retroceder (
            void )  [virtual]
```

Pone los motores marcha atrás.

Enciende los motores en sentido contrario a "avanzar".

Implements Puente_H.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/03-Puente_H/L298N.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/03-Puente_H/L298N.cpp

6.23 LCD Class Reference

Clase del objeto lcd El objeto lcd permite el manejo de displays digitales mediante comunicación de 4 bits.

`#include <LCD.h>`

Inheritance diagram for LCD:

Collaboration diagram for LCD:

Public Types

- enum {
  d7 = 0 , **d6** , **d5** , **d4** ,
  rs , **enable** }

Public Member Functions

- LCD (vector< gpio * > &salidas)
  Constructor de clase LCD.
- void Inicializar (const uint8_t filas, const uint8_t columnas)
  Inicializa el LCD.
- void Write (const char *s)
  Escribe en el LCD.
- void Write (const int32_t n)
  Escribe en el LCD un número.
- LCD & operator= (const char *s)
  Escribe en el LCD. Sobrecarga del operador =.
- void WriteAt (const int8_t *a, const uint8_t fila, const uint8_t columna)
  Escribe el LCD.
- void WriteAt (const int32_t n, const uint8_t fila, const uint8_t columna)
  Escribe el LCD.
- void Clear (void)
  Limpia el LCD.

Public Member Functions inherited from Display

- Display ()=default
- virtual void Write (const uint32_t n)=0
- virtual ~Display ()=default

Protected Member Functions

- void SWhandler (void)

  Ejecuta la instrucción del LCD periodicamente.

6.23.1 Detailed Description

Clase del objeto lcd El objeto lcd permite el manejo de displays digitales mediante comunicación de 4 bits.

6.23.2 Member Enumeration Documentation

6.23.2.1 anonymous enum

```
anonymous enum
```

Posiciones del buffer

6.23.3 Constructor & Destructor Documentation

6.23.3.1 LCD()

```
LCD::LCD (
            vector< gpio * > & salidas)
```

Constructor de clase LCD.

Crea LCD PIN con los parámetros correspondientes

Parameters

in	*salidas*	Vector de GPIO ordenado utilizadas para el LCD.

6.23.4 Member Function Documentation

6.23.4.1 Clear()

```
void LCD::Clear (
            void )  [virtual]
```

Limpia el LCD.

Deja el buffer como un string de caracteres ESPACIO que imprimen en blanco.

Implements Display.

6.23.4.2 Inicializar()

```
void LCD::Inicializar (
            const uint8_t filas,
            const uint8_t columnas)
```

Inicializa el LCD.

Crea el buffer y comienza a setear todas las salidas para comenzar a funcionar

Parameters

in	*filas*	Cantidad de filas del LCD.
in	*columnas*	Cantidad de columnas del LCD.

6.23.4.3 operator=()

```
LCD & LCD::operator= (
            const char * s)
```

Escribe en el LCD. Sobrecarga del operador =.

Escribe el buffer con el string indicado comenzando en la posición (0,0).

Parameters

in	*s*	String a escribir.

Returns

void

6.23.4.4 SWhandler()

```
void LCD::SWhandler (
            void )  [protected], [virtual]
```

Ejecuta la instrucción del LCD periodicamente.

Inicializa y escribe el LCD perdiódicamente utilizando el handler del systick.

Implements Callback.

6.23.4.5 Write() [1/2]

```
void LCD::Write (
            const char * s)
```

Escribe en el LCD.

Escribe el buffer con el string indicado comenzando en la posición (0,0).

Parameters

in	*s*	String a escribir.

6.23.4.6 Write() [2/2]

```
void LCD::Write (
            const int32_t n)
```

Escribe en el LCD un número.

Escribe el buffer con el string indicado comenzando en la posición (0,0). El número tendrá un máximo de 10 dígitos, sin contar el - de signo

Parameters

in	*n*	Número a escribir.

6.23.4.7 WriteAt() [1/2]

```
void LCD::WriteAt (
            const int32_t n,
            const uint8_t fila,
            const uint8_t columna)
```

Escribe el LCD.

Escribe en el buffer un número del LCD desde la posición indicada.

Parameters

in	*n*	numero a escribir.
in	*fila*	Fila donde empezar a escribir.
in	*columna*	Columna donde empezar a escribir.

6.23.4.8 WriteAt() [2/2]

```
void LCD::WriteAt (
            const int8_t * a,
            const uint8_t fila,
            const uint8_t columna)
```

Escribe el LCD.

Escribe en el buffer del LCD desde la posición indicada.

Parameters

in	*a*	string a escribir.
in	*fila*	Fila donde empezar a escribir.
in	*columna*	Columna donde empezar a escribir.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/LCD/LCD.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/LCD/LCD.cpp

6.24 MRThandler Class Reference

Clase del objeto MRThandler El objeto MRThandler debe ser heredado por cualquier objeto que desee estar conectado a las interrupciones del MRT timer.

```
#include <MRThandler.h>
```

Inheritance diagram for MRThandler:

Public Member Functions

- MRThandler (MRT_timer_channels _timer_number, MRT_MODES mode)

  Constructor de clase MRThandler.
- virtual ~MRThandler ()=default
- virtual void Handler (void)=0
- void **EneableInterrupt** (void)

  Activa la interrupcion.
- void **DisableInterrupt** (void)

  Desactiva la interrupcion.

Public Attributes

- const MRT_timer_channels m_timer_channel

Protected Member Functions

- void MRT_reset_time (void)

  Resetea el timer.
- uint32_t MRT_get_time (void)

  Devuelve el valor del timer en el momento actual.

6.24.1 Detailed Description

Clase del objeto MRThandler El objeto MRThandler debe ser heredado por cualquier objeto que desee estar conectado a las interrupciones del MRT timer.

6.24.2 Constructor & Destructor Documentation

6.24.2.1 MRThandler()

```
MRThandler::MRThandler (
            MRT_timer_channels _timer_number,
            MRT_MODES mode)
```

Constructor de clase MRThandler.

Crea un MRThandler con el canal correspondiente

Parameters

in	*_timer_number*	Canal del MRThandler.
in	*mode*	Modo del MRThandler.

6.24.2.2 ~MRThandler()

```
virtual MRThandler::~MRThandler ()  [virtual], [default]
```

Destructor por defecto

6.24.3 Member Function Documentation

6.24.3.1 Handler()

```
virtual void MRThandler::Handler (
            void )  [pure virtual]
```

Este método debe ser implementado por las clases derivadas cada una resolverá que hacer con su irq enganchada al systick del sistema

6.24.3.2 MRT_get_time()

```
uint32_t MRThandler::MRT_get_time (
            void )  [protected]
```

Devuelve el valor del timer en el momento actual.

Lee el registro del contador MRT del canal correspondiente.

Returns

devuelve el valor del timer.

6.24.3.3 MRT_reset_time()

```
void MRThandler::MRT_reset_time (
            void )  [protected]
```

Resetea el timer.

Vuelve a cargar el tiempo máximo en el registro.

6.24.4 Member Data Documentation

6.24.4.1 m_timer_channel

```
const MRT_timer_channels MRThandler::m_timer_channel
```

Canal del objeto MRT

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/09-MultiRateTimer/MRThandler.↩ h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/09-MultiRateTimer/MRThandler.cpp

6.25 Output Class Reference

Clase del objeto outputs.

`#include <Output.h>`

Inheritance diagram for Output:

Collaboration diagram for Output:

Public Member Functions

- Output (port_t puerto, uint8_t bit, mode_t modo, activity_t actividad=high, uint8_t estado=on)
  Constructor de clase Output.
- int8_t On (void)
  Funcion de encendido de la output.
- int8_t Off (void)
  Funcion de apagado de la output.
- int8_t SetUp (void)
  Funcion de configuracion del output.
- Output & operator= (uint8_t estado)
  Sobrecarga del operador =.
- bool operator== (uint8_t a)
  Sobrecarga del operador ==.
- void SWhandler (void) override
  Funcion de interrupcion del systick.
- virtual ~Output ()

Public Member Functions inherited from Callback

- void SetInterrupt ()
  Activa la interrupción.
- void UnSetInterrupt ()
  Desactiva la interrupción.

Additional Inherited Members

Protected Types inherited from gpio

- enum direction_t { **input** , **output** }
  Enumeracion de input/output.
- enum power_t { **off** , **on** }
  Enumeracion de encendido/apagado.
- enum mode_t {
  pushpull = 0 , **opencolector** , **inactive** = 0 , **pulldown** ,
  pullup , **repeater** }
  Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- enum activity_t { **low** , **high** }
  Enumeracion de activo bajo/alto.

- enum interrupt_mode_t { **rising_edge** = 0 , **falling_edge** , **rising_falling_edge** , **low_level** , **high_level** }

  Enumeracion de interrupciones del pin.
- typedef enum gpio::direction_t **direction_t**

  Enumeracion de input/output.
- typedef enum gpio::mode_t **mode_t**

  Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- typedef enum gpio::activity_t **activity_t**

  Enumeracion de activo bajo/alto.

Protected Types inherited from Pin

- enum port_t { **port0** , **port1** }
- enum max_bits_port_t { **b_port0** = 31 , **b_port1** = 9 }

  máximos pines por puerto
- enum error_t { **error** = 2 , **ok** }
- typedef enum Pin::port_t port_t
- typedef enum Pin::error_t error_t

Protected Member Functions inherited from gpio

- gpio (port_t port, uint8_t bit, mode_t mode, direction_t direction, activity_t activity=high)

  Constructor de clase GPIO.
- uint8_t SetPin (void) override

  Enciende la salida.
- uint8_t ClrPin (void) override

  Limpia el pin.
- uint8_t SetTogglePin (void) override

  Toggle del pin.
- uint8_t SetDir (void) override

  Setea la direccion.
- uint8_t SetToggleDir (void) override

  Toggle de la dirección.
- uint8_t GetPin (void) override

  Devuelve el valor del pin.
- uint8_t SetPinMode (void) override

  Configura el modo del pin.
- uint8_t SetPinResistor (void) override

  Setea la resistencia interna.
- gpio & operator= (uint8_t a)

  Sobrecarga del operador =.
- virtual ~gpio ()=default

Protected Member Functions inherited from InOut

- InOut ()=default
- virtual ~InOut ()=default

Protected Member Functions inherited from Pin

- Pin (port_t port, uint8_t bit)

  Constructor de clase PIN.

Protected Member Functions inherited from Callback

- void SetInterrupt ()

  Activa la interrupción.
- void UnSetInterrupt ()

  Desactiva la interrupción.

Protected Attributes inherited from gpio

- const mode_t m_mode
- direction_t m_direction
- const activity_t m_activity

Protected Attributes inherited from Pin

- const port_t m_port
- const uint8_t m_bit
- int8_t m_error

6.25.1 Detailed Description

Clase del objeto outputs.

El objeto outputs Permite el manejo de salidas de forma controlada con el systick y un buffer.

6.25.2 Constructor & Destructor Documentation

6.25.2.1 Output()

```
Output::Output (
            port_t puerto,
            uint8_t bit,
            mode_t modo,
            activity_t actividad = high,
            uint8_t estado = on)
```

Constructor de clase Output.

Crea un Output con los parámetros correspondientes.

Parameters

in	*puerto*	Puerto del objeto.
in	*bit*	Bit del objeto.
in	*modo*	Configuración eléctrica del pin.
in	*actividad*	Activo alto/bajo.
in	*estado*	Estado inicial.

6.25.2.2 ~Output()

```
Output::~Output ()  [virtual]
```

Destructor por defecto

6.25.3 Member Function Documentation

6.25.3.1 Off()

```
int8_t Output::Off (
            void )
```

Funcion de apagado de la output.

Guarda en el buffer el estado de apagado.

Returns

devuelve el error.

6.25.3.2 On()

```
int8_t Output::On (
            void )
```

Funcion de encendido de la output.

Guarda en el buffer el estado de encendido.

Returns

devuelve el error.

6.25.3.3 operator=()

```
Output & Output::operator= (
            uint8_t estado)
```

Sobrecarga del operador =.

Enciende la salida si se iguala a 1, apaga con 0.

Parameters

in	*estado*	Valor de igualacion.

Returns

Referencia a si mismo.

6.25.3.4 operator==()

```
bool Output::operator== (
            uint8_t a)
```

Sobrecarga del operador ==.

Indica si la salida se encuentra en el estado "a" o no.

Parameters

in	*a*	Valor a comparar el buffer.

Returns

verdadero o falso.

6.25.3.5 SetUp()

```
int8_t Output::SetUp (
            void )
```

Funcion de configuracion del output.

Setea la direccion y resistencia de la salida.

Returns

devuelve el error.

6.25.3.6 SWhandler()

```
void Output::SWhandler (
            void )  [override], [virtual]
```

Funcion de interrupcion del systick.

Coloca la salida con el valor del buffer.

Implements Callback.

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/05-Output/Output.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/05-Output/Output.cpp

6.26 Pin Class Reference

Clase del objeto Pin.

```
#include <Pin.h>
```

Inheritance diagram for Pin:

Public Types

- enum port_t { **port0** , **port1** }
- enum max_bits_port_t { **b_port0** = 31 , **b_port1** = 9 }

  máximos pines por puerto
- enum error_t { **error** = 2 , **ok** }
- typedef enum Pin::port_t port_t
- typedef enum Pin::error_t error_t

Public Member Functions

- Pin (port_t port, uint8_t bit)

  Constructor de clase PIN.

Public Attributes

- const port_t m_port
- const uint8_t m_bit
- int8_t m_error

6.26.1 Detailed Description

Clase del objeto Pin.

El objeto Pin debe ser heredado y posee el puerto y bit del periférico en cuestion

6.26.2 Member Typedef Documentation

6.26.2.1 error_t

```
typedef enum Pin::error_t Pin::error_t
```

Enumeracion de error

6.26.2.2 port_t

```
typedef enum Pin::port_t Pin::port_t
```

Enumeracion con los puertos

6.26.3 Member Enumeration Documentation

6.26.3.1 error_t

```
enum Pin::error_t
```

Enumeracion de error

6.26.3.2 port_t

```
enum Pin::port_t
```

Enumeracion con los puertos

6.26.4 Constructor & Destructor Documentation

6.26.4.1 Pin()

```
Pin::Pin (
            port_t port,
            uint8_t bit)
```

Constructor de clase PIN.

Crea un PIN con los parámetros correspondientes

Parameters

in	*port*	Puerto del objeto
in	*bit*	Bit del objeto

6.26.5 Member Data Documentation

6.26.5.1 m_bit

```
const uint8_t Pin::m_bit
```

Guarda el bit del pin

6.26.5.2 m_error

```
int8_t Pin::m_error
```

Error en la clase pin

6.26.5.3 m_port

```
const port_t Pin::m_port
```

Guarda el puerto del pin

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/01-Pin/Pin.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/01-Pin/Pin.cpp

6.27 PinInterrupt Class Reference

Clase del objeto Pin_interrupt El objeto Pin_interrupt debe ser heredado por cualquier objeto que desee tener interupciones por pin.

`#include <Pininterrupt.h>`

Inheritance diagram for PinInterrupt:

Collaboration diagram for PinInterrupt:

Public Member Functions

- PinInterrupt (port_t port, uint8_t bit, mode_t gpio_mode, activity_t activity, uint8_t intrp_mode)
  Constructor de clase Pin_interrupt.
- void EnableInterupt (void)
  Habilito la interrupción.
- void DisableInterupt (void)
  Deshabilito la interrupción.
- void PinInterrupt_Inicializar (void)
  Inicializo la interrupción por pin.
- virtual void GpioHandler (void)=0
- virtual ~PinInterrupt ()

Public Attributes

- const uint8_t m_interrupt_number
- const uint8_t m_interrput_mode

Static Public Attributes

- static uint8_t m_cant = 0

Additional Inherited Members

Protected Types inherited from gpio

- enum direction_t { **input** , **output** }
  Enumeracion de input/output.
- enum power_t { **off** , **on** }
  Enumeracion de encendido/apagado.
- enum mode_t {
  pushpull = 0 , **opencolector** , **inactive** = 0 , **pulldown** ,
  pullup , **repeater** }
  Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- enum activity_t { **low** , **high** }
  Enumeracion de activo bajo/alto.
- enum interrupt_mode_t {
  rising_edge = 0 , **falling_edge** , **rising_falling_edge** , **low_level** ,
  high_level }
  Enumeracion de interrupciones del pin.
- typedef enum gpio::direction_t **direction_t**
  Enumeracion de input/output.
- typedef enum gpio::mode_t **mode_t**
  Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- typedef enum gpio::activity_t **activity_t**
  Enumeracion de activo bajo/alto.

Protected Types inherited from Pin

- enum port_t { **port0** , **port1** }
- enum max_bits_port_t { **b_port0** = 31 , **b_port1** = 9 }

  máximos pines por puerto
- enum error_t { **error** = 2 , **ok** }
- typedef enum Pin::port_t port_t
- typedef enum Pin::error_t error_t

Protected Member Functions inherited from gpio

- gpio (port_t port, uint8_t bit, mode_t mode, direction_t direction, activity_t activity=high)

  Constructor de clase GPIO.
- uint8_t SetPin (void) override

  Enciende la salida.
- uint8_t ClrPin (void) override

  Limpia el pin.
- uint8_t SetTogglePin (void) override

  Toggle del pin.
- uint8_t SetDir (void) override

  Setea la direccion.
- uint8_t SetToggleDir (void) override

  Toggle de la dirección.
- uint8_t GetPin (void) override

  Devuelve el valor del pin.
- uint8_t SetPinMode (void) override

  Configura el modo del pin.
- uint8_t SetPinResistor (void) override

  Setea la resistencia interna.
- gpio & operator= (uint8_t a)

  Sobrecarga del operador =.
- virtual ~gpio ()=default

Protected Member Functions inherited from InOut

- InOut ()=default
- virtual ~InOut ()=default

Protected Member Functions inherited from Pin

- Pin (port_t port, uint8_t bit)

  Constructor de clase PIN.

Protected Attributes inherited from gpio

- const mode_t m_mode
- direction_t m_direction
- const activity_t m_activity

Protected Attributes inherited from Pin

- const port_t m_port
- const uint8_t m_bit
- int8_t m_error

6.27.1 Detailed Description

Clase del objeto Pin_interrupt El objeto Pin_interrupt debe ser heredado por cualquier objeto que desee tener interupciones por pin.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 PinInterrupt()

```
PinInterrupt::PinInterrupt (
            port_t port,
            uint8_t bit,
            mode_t gpio_mode,
            activity_t activity,
            uint8_t intrp_mode)
```

Constructor de clase Pin_interrupt.

Crea un Pin_interrupt con los parámetros correspondientes

Parameters

in	*port*	Puerto del pin interrutp.
in	*bit*	bit del pin interrutp.
in	*gpio_mode*	Configuración eléctrica del pin.
in	*activity*	Activo alto/bajo.
in	*intrp_mode*	Tipo de interrupción.

6.27.2.2 ~PinInterrupt()

```
PinInterrupt::~PinInterrupt ()  [virtual]
```

Destructor por defecto

6.27.3 Member Function Documentation

6.27.3.1 DisableInterupt()

```
void PinInterrupt::DisableInterupt (
            void )
```

Deshabilito la interrupción.

Deshabilito la interrupción de la pata específica que estoy utilizando.

6.27.3.2 EnableInterupt()

```
void PinInterrupt::EnableInterupt (
            void )
```

Habilito la interrupción.

Habilito la interrupción de la pata específica que estoy utilizando.

6.27.3.3 GpioHandler()

```
virtual void PinInterrupt::GpioHandler (
            void )  [pure virtual]
```

Funcion Handler que se ejecuta al realizarse la interrupcion

Implemented in PWM_Reader.

6.27.3.4 PinInterrupt_Inicializar()

```
void PinInterrupt::PinInterrupt_Inicializar (
            void )
```

Inicializo la interrupción por pin.

Modifico todos los registros para que la interrupción por pin esté configurada.

6.27.4 Member Data Documentation

6.27.4.1 m_cant

```
uint8_t PinInterrupt::m_cant = 0  [static]
```

Cantidad de PIN INTERRUPT creados globalmente

6.27.4.2 m_interrput_mode

```
const uint8_t PinInterrupt::m_interrput_mode
```

Tipo de interrupción del pin

6.27.4.3 m_interrupt_number

```
const uint8_t PinInterrupt::m_interrupt_number
```

Número de PIN INTERRUPT

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/03-PinInterrupt/Pininterrupt.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/03-PinInterrupt/Pininterrupt.cpp

6.28 Puente_H Class Reference

Clase del objeto Puente_H El objeto Puente_H es la interfaz abstracta pura de cualquier puente H que se desee realizar.

`#include <PuenteH.h>`

Inheritance diagram for Puente_H:

Public Types

- enum { **IZQUIERDA** = 0 , **DERECHA** }

Public Member Functions

- Puente_H ()=default
- virtual void Inicializar (void)=0
- virtual void GirarIzq (void)=0
- virtual void GirarDer (void)=0
- virtual void Girar (uint8_t direccion)=0
- virtual void Frenar (void)=0
- virtual void Avanzar (void)=0
- virtual void Retroceder (void)=0
- virtual ~Puente_H ()=default

6.28.1 Detailed Description

Clase del objeto Puente_H El objeto Puente_H es la interfaz abstracta pura de cualquier puente H que se desee realizar.

6.28.2 Member Enumeration Documentation

6.28.2.1 anonymous enum

`anonymous enum`

Enumeracion de sentidos de giro

6.28.3 Constructor & Destructor Documentation

6.28.3.1 Puente_H()

`Puente_H::Puente_H () [default]`

Constructor por defecto

6.28.3.2 ~Puente_H()

```
virtual Puente_H::~Puente_H ()  [virtual], [default]
```

Destructor por defecto

6.28.4 Member Function Documentation

6.28.4.1 Avanzar()

```
virtual void Puente_H::Avanzar (
            void )  [pure virtual]
```

Avanza los motores

Implemented in L298N.

6.28.4.2 Frenar()

```
virtual void Puente_H::Frenar (
            void )  [pure virtual]
```

Frena los motores

Implemented in L298N.

6.28.4.3 Girar()

```
virtual void Puente_H::Girar (
            uint8_t direccion)  [pure virtual]
```

Gira los motores

Implemented in L298N.

6.28.4.4 GirarDer()

```
virtual void Puente_H::GirarDer (
            void )  [pure virtual]
```

Gira a la derecha

Implemented in L298N.

6.28.4.5 GirarIzq()

```
virtual void Puente_H::GirarIzq (
            void )  [pure virtual]
```

Gira a la izquierda

Implemented in L298N.

6.28.4.6 Inicializar()

```
virtual void Puente_H::Inicializar (
            void )  [pure virtual]
```

Inicializa el Puente H

Implemented in L298N.

6.28.4.7 Retroceder()

```
virtual void Puente_H::Retroceder (
            void )  [pure virtual]
```

Retrocede los motores

Implemented in L298N.

The documentation for this class was generated from the following file:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/03-Puente_H/PuenteH.h

6.29 Pwm Class Reference

Clase del objeto Pwm.

`#include <Pwm.h>`

Inheritance diagram for Pwm:

Collaboration diagram for Pwm:

Public Types

- enum pwm_time_unit_t { **SEG** = 0 , **MILI_SEG** , **MICRO_SEG** }
- enum pwm_channel_t {
  CHANNEL_1 = 1 , **CHANNEL_2** , **CHANNEL_3** , **CHANNEL_4** ,
  CHANNEL_5 , **CHANNEL_6** }
- enum activity_t { **low** , **high** }

Public Member Functions

- Pwm (port_t puerto, uint8_t bit, uint8_t actividad, pwm_channel_t number)
  Constructor de clase Pwm.
- void Inicializar (uint32_t ton, uint32_t toff, pwm_time_unit_t t=MICRO_SEG)
  Inicializo el PWM.
- void SetTon (uint32_t time, pwm_time_unit_t t=MICRO_SEG)
  Seteo el tiempo de encendido del PWM.
- void SetPeriod (uint32_t time, pwm_time_unit_t t=MICRO_SEG)
  Seteo el periodo del PWM.
- void On ()
  Enciende el PWM.
- void Off ()
  Apaga el PWM.
- virtual ~Pwm ()

Protected Attributes

- const uint8_t m_activity
- uint32_t m_ton
- uint32_t m_toff
- const uint8_t m_pwm_channel

Protected Attributes inherited from Pin

- const port_t m_port
- const uint8_t m_bit
- int8_t m_error

Additional Inherited Members

Protected Types inherited from Pin

- enum port_t { **port0** , **port1** }
- enum max_bits_port_t { **b_port0** = 31 , **b_port1** = 9 }

  máximos pines por puerto
- enum error_t { **error** = 2 , **ok** }
- typedef enum Pin::port_t port_t
- typedef enum Pin::error_t error_t

Protected Member Functions inherited from SCtimer

- **SCtimer** ()

  Constructor de clase SCtimer.
- void SetTime (uint32_t time, uint32_t channel)

  Setea el tiempo de un canal del contador.
- void SetUnify (bool a)

  Unifica los dos registros del SCT.
- void SetAutoLimit (bool a)

  Setea el autolimite.
- void SetSwitchMatrizSCTOUT (uint8_t bit, uint8_t port, uint8_t out_number)

  Configura la switch Matrix.
- void StartTimer (void)

  Habilita el timer.
- void StopTimer (void)

  Deshabilita el timer.
- virtual ~SCtimer ()

Protected Member Functions inherited from Pin

- Pin (port_t port, uint8_t bit)

  Constructor de clase PIN.

6.29.1 Detailed Description

Clase del objeto Pwm.

FUNCIONAMIENTO: Genera un PWM realizando un Match entre distintos eventos y señales del SCTimer. Por cómo está configurado, todos los objetos PWM tendrán el mismo periodo. Esto se debe a que todos funcionan con el Evento 0 que los resetea. Debido a esto, solo se puede tener hasta 6 PWM al mismo tiempo y todos tendrán el mismo PERIODO.

6.29.2 Member Enumeration Documentation

6.29.2.1 activity_t

```
enum Pwm::activity_t
```

Enum de activo alto/bajo

6.29.2.2 pwm_channel_t

```
enum Pwm::pwm_channel_t
```

Canales del PWM

6.29.2.3 pwm_time_unit_t

```
enum Pwm::pwm_time_unit_t
```

Unidad de tiempo del PWM

6.29.3 Constructor & Destructor Documentation

6.29.3.1 Pwm()

```
Pwm::Pwm (
            port_t puerto,
            uint8_t bit,
            uint8_t actividad,
            pwm_channel_t number)
```

Constructor de clase Pwm.

Crea un Pwm con los parámetros correspondientes

Parameters

in	*puerto*	Puerto del PWM.
in	*bit*	Bit del PWM.
in	*actividad*	Activo alto/bajo.
in	*number*	Canal del SCtimer a utilizar.

6.29.3.2 ~Pwm()

```
Pwm::~Pwm ()  [virtual]
```

Destructor por defecto

6.29.4 Member Function Documentation

6.29.4.1 Inicializar()

```
void Pwm::Inicializar (
            uint32_t ton,
            uint32_t toff,
            pwm_time_unit_t t = MICRO_SEG)
```

Inicializo el PWM.

Utilizando los regristros configuro todo para la utilizacion del PWM.

Parameters

in	*ton*	Tiempo de encendido.
in	*toff*	Tiempo de apagado (no es el periodo. El periodo es la suma de ambos).
in	*t*	Unidad de medida de los tiempos de encendido y apagado.

6.29.4.2 Off()

```
void Pwm::Off (
            void )
```

Apaga el PWM.

Utiliza los registros con StopTimer para deshabilitar la salida.

6.29.4.3 On()

```
void Pwm::On (
            void )
```

Enciende el PWM.

Utiliza los registros con StarTimer para habilitar la salida.

6.29.4.4 SetPeriod()

```
void Pwm::SetPeriod (
            uint32_t time,
            pwm_time_unit_t t = MICRO_SEG)
```

Seteo el periodo del PWM.

Utilizando los regristros con SetTime creo el tiempo de apagado de mi PWM.

Parameters

in	*time*	Tiempo del periodo.
in	*t*	Unidad de medida del tiempo de periodo.

6.29.4.5 SetTon()

```
void Pwm::SetTon (
            uint32_t time,
            pwm_time_unit_t t = MICRO_SEG)
```

Seteo el tiempo de encendido del PWM.

Utilizando los regristros con SetTime creo el tiempo de encendido de mi PWM.

Parameters

in	*time*	Tiempo de encendido.
in	*t*	Unidad de medida del tiempo de encendido.

6.29.5 Member Data Documentation

6.29.5.1 m_activity

```
const uint8_t Pwm::m_activity  [protected]
```

Activo alto/bajo del PWM

6.29.5.2 m_pwm_channel

```
const uint8_t Pwm::m_pwm_channel  [protected]
```

Canal del objeto PWM

6.29.5.3 m_toff

```
uint32_t Pwm::m_toff  [protected]
```

Tiempo de apagado del PWM en micro segundos

6.29.5.4 m_ton

```
uint32_t Pwm::m_ton  [protected]
```

Tiempo de encendido del PWM en micro segundos

The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/05-PWM/Pwm.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/05-PWM/Pwm.cpp

6.30 PWM_Reader Class Reference

Clase del objeto PWM_Reader.

`#include <PWMReader.h>`

Inheritance diagram for PWM_Reader:

Collaboration diagram for PWM_Reader:

Public Member Functions

- PWM_Reader (port_t puerto, uint8_t bit, mode_t modo, activity_t activity, MRT_timer_channels timer_channel)

  Constructor de clase PWM_Reader.
- void Inicializar (void)

  Inicializo el PWM_In.
- uint32_t GetPulseOn (void) const

  Obtengo el largo del pulso.
- void Off (void)

  Apago el PWM_In.
- void On (void)

  Enciendo el PWM_In.

Protected Member Functions

- void GpioHandler (void)

  Handler de la interrupción por flanco.

Protected Member Functions inherited from PinInterrupt

- PinInterrupt (port_t port, uint8_t bit, mode_t gpio_mode, activity_t activity, uint8_t intrp_mode)

  Constructor de clase Pin_interrupt.
- void EnableInterupt (void)

  Habilito la interrupción.
- void DisableInterupt (void)

  Deshabilito la interrupción.
- void PinInterrupt_Inicializar (void)

  Inicializo la interrupción por pin.
- virtual ~PinInterrupt ()

Protected Member Functions inherited from gpio

- gpio (port_t port, uint8_t bit, mode_t mode, direction_t direction, activity_t activity=high)

  Constructor de clase GPIO.
- uint8_t SetPin (void) override

  Enciende la salida.
- uint8_t ClrPin (void) override

  Limpia el pin.
- uint8_t SetTogglePin (void) override

  Toggle del pin.
- uint8_t SetDir (void) override

  Setea la direccion.
- uint8_t SetToggleDir (void) override

  Toggle de la dirección.
- uint8_t GetPin (void) override

  Devuelve el valor del pin.
- uint8_t SetPinMode (void) override

  Configura el modo del pin.
- uint8_t SetPinResistor (void) override

  Setea la resistencia interna.
- gpio & operator= (uint8_t a)

  Sobrecarga del operador =.
- virtual ~gpio ()=default

Protected Member Functions inherited from InOut

- InOut ()=default
- virtual ~InOut ()=default

Protected Member Functions inherited from Pin

- Pin (port_t port, uint8_t bit)

  Constructor de clase PIN.

Protected Member Functions inherited from MRThandler

- void MRT_reset_time (void)

  Resetea el timer.
- uint32_t MRT_get_time (void)

  Devuelve el valor del timer en el momento actual.

- MRThandler (MRT_timer_channels _timer_number, MRT_MODES mode)

  Constructor de clase MRThandler.
- virtual ~MRThandler ()=default
- virtual void Handler (void)=0
- void **EneableInterrupt** (void)

  Activa la interrupcion.
- void **DisableInterrupt** (void)

  Desactiva la interrupcion.

Additional Inherited Members

Protected Types inherited from gpio

- enum direction_t { **input** , **output** }

  Enumeracion de input/output.
- enum power_t { **off** , **on** }

  Enumeracion de encendido/apagado.
- enum mode_t {
  pushpull = 0 , **opencolector** , **inactive** = 0 , **pulldown** ,
  pullup , **repeater** }

  Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- enum activity_t { **low** , **high** }

  Enumeracion de activo bajo/alto.
- enum interrupt_mode_t {
  rising_edge = 0 , **falling_edge** , **rising_falling_edge** , **low_level** ,
  high_level }

  Enumeracion de interrupciones del pin.
- typedef enum gpio::direction_t **direction_t**

  Enumeracion de input/output.
- typedef enum gpio::mode_t **mode_t**

  Enumeracion con los modos de la salida. Resaltar que los primeros son de OUTPUT y los segundos de INPUT.
- typedef enum gpio::activity_t **activity_t**

  Enumeracion de activo bajo/alto.

Protected Types inherited from Pin

- enum port_t { **port0** , **port1** }
- enum max_bits_port_t { **b_port0** = 31 , **b_port1** = 9 }

  máximos pines por puerto
- enum error_t { **error** = 2 , **ok** }
- typedef enum Pin::port_t port_t
- typedef enum Pin::error_t error_t

Protected Attributes inherited from PinInterrupt

- const uint8_t m_interrupt_number
- const uint8_t m_interrput_mode

Protected Attributes inherited from gpio

- const mode_t m_mode
- direction_t m_direction
- const activity_t m_activity

Protected Attributes inherited from Pin

- const port_t m_port
- const uint8_t m_bit
- int8_t m_error

Protected Attributes inherited from MRThandler

- const MRT_timer_channels m_timer_channel

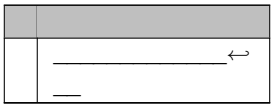
Static Protected Attributes inherited from PinInterrupt

- static uint8_t m_cant = 0

6.30.1 Detailed Description

Clase del objeto PWM_Reader.

Lee ancho de señal de una entrada en milisegundos. Solo lee pulsos completos. Si se queda en 0, no lo leerá hasta volver a 1. Lectura:



|—m_pulse_on—| |—m_pulse_on----------—

6.30.2 Constructor & Destructor Documentation

6.30.2.1 PWM_Reader()

```
PWM_Reader::PWM_Reader (
            port_t puerto,
            uint8_t bit,
            mode_t modo,
            activity_t activity,
            MRT_timer_channels timer_channel)
```

Constructor de clase PWM_Reader.
Crea un PWM_Reader con los parámetros correspondientes

Parameters

in	*puerto*	Puerto del PWM_Reader.
in	*bit*	Bit del PWM_Reader.
in	*modo*	Configuracion eléctrica de la entrada.
in	*activity*	activo alto/bajo.
in	*timer_channel*	Canal del MRTtimer a utilizar.

6.30.3 Member Function Documentation

6.30.3.1 GetPulseOn()

```
uint32_t PWM_Reader::GetPulseOn (
            void ) const
```

Obtengo el largo del pulso.
Realizo la cuenta del tiempo transcurrido entre el último flanco y este.

Returns

largo del pulso

6.30.3.2 GpioHandler()

```
void PWM_Reader::GpioHandler (
            void )  [protected], [virtual]
```

Handler de la interrupción por flanco.
Destructor por defecto
Si el flanco es ascendente reseteo el contador. Si es desendente guado el valor del contador. La cuenta se realiza en otra función para ahorrar tiempo acá
Implements PinInterrupt.

6.30.3.3 Inicializar()

```
void PWM_Reader::Inicializar (
            void )
```

Inicializo el PWM_In.
Seteo dirección resistencia y habilito la interrupción por flancos.

6.30.3.4 Off()

```
void PWM_Reader::Off (
            void )
```

Apago el PWM_In.
Deshabilito la interrupción y seteo el pulso a un valor gigante que no genere overflow.

6.30.3.5 On()

```
void PWM_Reader::On (
            void )
```

Enciendo el PWM_In.
Habilito la interrupción.
The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/06-PWM_Reader/PWMReader.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/06-PWM_Reader/PWMReader.↩ cpp

6.31 Reloj Class Reference

Clase del objeto Reloj.
`#include <Reloj.h>`
Inheritance diagram for Reloj:
Collaboration diagram for Reloj:

Public Member Functions

- Reloj ()

  Constructor de clase Reloj.
- int32_t GetHour (void) const

  Devuelve el valor de hora.
- int32_t GetMin (void) const

  Devuelve el valor de minutos.
- int32_t GetSeg (void) const

  Devuelve el valor de segundos.
- void Reset (void)

  Resetea el reloj.
- void SetTime (const int32_t _hour, const int32_t _min=-1, const int32_t _seg=-1)

  Setea el reloj.

Public Member Functions inherited from Callback

- void SetInterrupt ()

  Activa la interrupción.
- void UnSetInterrupt ()

  Desactiva la interrupción.

Protected Member Functions

- void SWhandler (void)

  Actualiza el reloj.

6.31.1 Detailed Description

Clase del objeto Reloj.
El objeto Reloj se comportará como un reloj que iniciará a contar desde 0hs o desde el valor asignado por el usuario.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 Reloj()

```
Reloj::Reloj ()
```

Constructor de clase Reloj.
Crea un Reloj por defecto

6.31.3 Member Function Documentation

6.31.3.1 GetHour()

```
int32_t Reloj::GetHour (
            void ) const
```

Devuelve el valor de hora.
Entrega la variable m_hora.

Returns

m_hora

6.31.3.2 GetMin()

```
int32_t Reloj::GetMin (
            void ) const
```

Devuelve el valor de minutos.
Entrega la variable m_minutos.

Returns

m_minutos

6.31.3.3 GetSeg()

```
int32_t Reloj::GetSeg (
            void ) const
```

Devuelve el valor de segundos.
Entrega la variable m_segundos.

Returns

m_segundos

6.31.3.4 Reset()

```
void Reloj::Reset (
            void )
```

Resetea el reloj.
Coloca todas las variables en 0.

6.31.3.5 SetTime()

```
void Reloj::SetTime (
            const int32_t _hour,
            const int32_t _min = -1,
            const int32_t _seg = -1)
```

Setea el reloj.

Parameters

in	*_hour*	horas a setear.
in	*_min*	minutos a setear.
in	*_seg*	segundos a setear.

Configura el valor actual de horas, minutos y segundos.

6.31.3.6 SWhandler()

```
void Reloj::SWhandler (
            void )  [protected], [virtual]
```

Actualiza el reloj.
Cada 1000 ticks pasa un segundo. Lo suma a la cuenta.
Implements Callback.
The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/04-Reloj/Reloj.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/04-Reloj/Reloj.cpp

6.32 SCtimer Class Reference

Clase del objeto SCtimer El objeto SCtimer debe ser heredado por quienes desen utilizar las interrucpiones o funcionalidades del SCtimer.
`#include <SCtimer.h>`
Inheritance diagram for SCtimer:

Public Member Functions

- **SCtimer** ()

  Constructor de clase SCtimer.
- void SetTime (uint32_t time, uint32_t channel)

  Setea el tiempo de un canal del contador.
- void SetUnify (bool a)

  Unifica los dos registros del SCT.
- void SetAutoLimit (bool a)

  Setea el autolimite.
- void SetSwitchMatrizSCTOUT (uint8_t bit, uint8_t port, uint8_t out_number)

  Configura la switch Matrix.
- void StartTimer (void)

  Habilita el timer.

- void StopTimer (void)

  Deshabilita el timer.
- virtual ~SCtimer ()

6.32.1 Detailed Description

Clase del objeto SCtimer El objeto SCtimer debe ser heredado por quienes desen utilizar las interrucpiones o funcionalidades del SCtimer.

6.32.2 Constructor & Destructor Documentation

6.32.2.1 ~SCtimer()

```
SCtimer::~SCtimer ()  [virtual]
```

Destructor por defecto

6.32.3 Member Function Documentation

6.32.3.1 SetAutoLimit()

```
void SCtimer::SetAutoLimit (
            bool a)
```

Setea el autolimite.
Determina si se resetea todos los canales al activar el canal 0 o no.

Parameters

in	*a*	bool que indica si el autolimite es cierto o falso.

6.32.3.2 SetSwitchMatrizSCTOUT()

```
void SCtimer::SetSwitchMatrizSCTOUT (
            uint8_t bit,
            uint8_t port,
            uint8_t out_number)
```

Configura la switch Matrix.
Configura en la switch matrix la salida del evento out_number al puerto y pin indicados.

Parameters

in	*bit*	bit de la salida a programar.
in	*port*	puerto de la salida a programar.
in	*out_number*	número de evento al que corresponderán el bit y puerto.

6.32.3.3 SetTime()

```
void SCtimer::SetTime (
            uint32_t time,
            uint32_t channel)
```

Setea el tiempo de un canal del contador.
Configura un canal para que active dentro de determinado tiempo.

Parameters

in	*time*	tiempo de la acción.
in	*channel*	canal a configurar.

6.32.3.4 SetUnify()

```
void SCtimer::SetUnify (
            bool a)
```

Unifica los dos registros del SCT.
Elije si trabajar con registros high y low o con uno solo.

Parameters

in	*a*	bool que indica si la unificación es cierta o falsa.

6.32.3.5 StartTimer()

```
void SCtimer::StartTimer (
            void )
```

Habilita el timer.
Enciende el clock del timer.

6.32.3.6 StopTimer()

```
void SCtimer::StopTimer (
            void )
```

Deshabilita el timer.
Apaga el clock del timer.
The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/10-SCTimer/SCtimer.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/10-SCTimer/SCtimer.cpp

6.33 segmentos Class Reference

Clase del objeto segmentos Clase abstracta pura para la generación de segmentos.
`#include <Segmentos.h>`
Inheritance diagram for segmentos:

Public Member Functions

- virtual void SetSegmentos (uint16_t)=0
- virtual void Inicializar (void)=0

6.33.1 Detailed Description

Clase del objeto segmentos Clase abstracta pura para la generación de segmentos.

6.33.2 Member Function Documentation

6.33.2.1 Inicializar()

```
virtual void segmentos::Inicializar (
            void )  [pure virtual]
```

Funcion de inicializacion
Implemented in I4511.

6.33.2.2 SetSegmentos()

```
virtual void segmentos::SetSegmentos (
            uint16_t )  [pure virtual]
```

constructor por defecto Funcion set del segmento
Implemented in I4511.
The documentation for this class was generated from the following file:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/B-↩ Segmentos/Segmentos.h

6.34 SPI Class Reference

Inheritance diagram for SPI:
Collaboration diagram for SPI:

Public Types

- enum **SPI_mode_t** { **master** = 1 , **slave** = 2 }

Public Member Functions

- SPI ()

  Constructor de la clase I2C.
- void **Initialize** (uint32_t clk_freq)
- void **EnableInterupt** (void)
- void **DisableInterupt** (void)
- void Write (uint8_t data)
- int8_t **Read** (uint8_t *data, bool continue_reading)
- virtual void SPI_IRQHandler (void)

6.34.1 Constructor & Destructor Documentation

6.34.1.1 SPI()

```
SPI::SPI ()
```

Constructor de la clase I2C.
Genera un I2C con los parametros indicados

6.34.2 Member Function Documentation

6.34.2.1 SPI_IRQHandler()

```
virtual void SPI::SPI_IRQHandler (
            void )  [inline], [virtual]
```

Handler generico de interrupcion I2C. No hace nada, debe heredarse y sobreescribirse.

6.34.2.2 Write()

```
void SPI::Write (
            uint8_t data)  [virtual]
```

Funcion de escritura
Implements ComunicacionSincronica.
The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/14-ComSincronica/SPI/SPI.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/14-ComSincronica/SPI/SPI.cpp

6.35 teclado Class Reference

Clase del objeto teclado FUNCIONAMIENTO: Este objeto permite controlar teclados matriciales cableados, eliminando el rebote mecánico. Teclado de tipo mono-usuario, mono-dedo con opción de mantener presionado una tecla.

`#include <teclado.h>`

Inheritance diagram for teclado:

Collaboration diagram for teclado:

Public Member Functions

- teclado (vector< gpio * > &s, vector< gpio * > &r)

  Constructor de un teclado.
- void SWhandler (void)

  Funcion handler/interrupcion del teclado.
- void Inicializar (void)

  Inicializa el teclado.
- uint8_t Get (void)

  Devuelve el valor presionado en el teclado.
- virtual ~teclado ()

Public Member Functions inherited from Callback

- void SetInterrupt ()

  Activa la interrupción.
- void UnSetInterrupt ()

  Desactiva la interrupción.

6.35.1 Detailed Description

Clase del objeto teclado FUNCIONAMIENTO: Este objeto permite controlar teclados matriciales cableados, eliminando el rebote mecánico. Teclado de tipo mono-usuario, mono-dedo con opción de mantener presionado una tecla.

6.35.2 Constructor & Destructor Documentation

6.35.2.1 teclado()

```
teclado::teclado (
            vector< gpio * > & s,
            vector< gpio * > & r)
```

Constructor de un teclado.

Construye un objeto teclado con los parámetros indicados

Parameters

in	*s*	vector de filas
in	*r*	Vector de columnas

6.35.2.2 ~teclado()

```
teclado::~teclado ()  [virtual]
```

Destructor por defecto

6.35.3 Member Function Documentation

6.35.3.1 Get()

```
uint8_t teclado::Get (
            void )
```

Devuelve el valor presionado en el teclado.
Entrega el valor del buffer o NO_KEY en caso de no entregar nada. La tecla va desde 0 hasta el (filas∗columnas)

Returns

Tecla presionada

6.35.3.2 Inicializar()

```
void teclado::Inicializar (
            void )
```

Inicializa el teclado.
Setea las direcciones y resistencias de las distintas entradas y salidas GPIO

6.35.3.3 SWhandler()

```
void teclado::SWhandler (
            void )  [virtual]
```

Funcion handler/interrupcion del teclado.
Ejecuta el barrido y la lectura del antirrebote
Implements Callback.
The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/01-Teclado/teclado.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/01-Teclado/teclado.cpp

6.36 Timer Class Reference

Clase del objeto timer.
`#include <Timer.h>`
Inheritance diagram for Timer:
Collaboration diagram for Timer:

Public Types

- enum bases_t { **DEC** , **SEG** , **MIN** , **HOR** }
- enum erroresTimers_t { **errorTimer** , **OKtimers** }

  Enumeración de error del timer.
- enum standby_t { **RUN** , **PAUSE** }

  Enumeracion de stand By.
- typedef enum Timer::bases_t bases_t

Public Member Functions

- Timer ()

  Constructor de clase timer.
- Timer (const bases_t base, const Timer_Handler handler=nullptr)

  Constructor de clase timer.
- void TimerStart (uint32_t time, const Timer_Handler handler, const bases_t base)

  Inicia un timer.
- void SetTimer (uint32_t time)

  Inicia un timer.

- uint32_t GetTimer (void) const

  Toma el valor al vuelo del timer en cuestion.
- void StandByTimer (const uint8_t accion)

  Detiene/Arranca el timer, NO lo resetea.
- void SetTimerBase (const bases_t base)

  Setea la base de tiempo.
- void **TimerStop** (void)

  Detiene el timer.
- uint32_t GetTmrRun (void)

  Obtiene el valor del timer.
- void **SetTmrEvent** (void)

  Setea el evento del timer.
- void **ClrTmrEvent** (void)

  Limpia el evento del timer.
- bool GetTmrEvent (void)

  Indica si el timer venció o no.
- bool GetmrStandBy (void)

  Informa si el timer está detenido o no.
- void SetmrStandBy (uint8_t accion)

  Detiene el timer sin apagarlo.
- void SetTmrHandler (void)

  Ejecuta el timer Handler.
- void TimerStart (uint32_t time)

  Inicia un timer.
- Timer & operator= (uint32_t t)

  Sobrecarga de del operador de asignacion.
- bool operator! ()

  Sobrecarga de del operador de negacion.
- operator bool ()

  Sobrecarga de del operador de contenido.
- bool operator== (uint32_t t)

  Sobrecarga de del operador de comparacion.
- void SWhandler (void)

  Decremento periodico del timer. Debe ser llamada periodicamente con la base de tiempos.
- int8_t TmrEvent (void)

  Función encargada de revisar si los timer vencieron y ejecuta automáticamente su función asignada.

Public Member Functions inherited from Callback

- void SetInterrupt ()

  Activa la interrupción.
- void UnSetInterrupt ()

  Desactiva la interrupción.

Protected Attributes

- volatile uint32_t m_TmrRun
- volatile bool m_TmrEvent
- void(* m_TmrHandler)(void)
- volatile bool m_TmrStandBy
- volatile bases_t m_TmrBase

Friends

- bool operator== (uint32_t t, Timer &T)

  Sobrecarga de del operador de comparacion.

6.36.1 Detailed Description

Clase del objeto timer.
El objeto timer se conecta al systick y ejecuta una "alarma" al terminar el tiempo. Se comporta como un cronómetro o contador de microondas.

6.36.2 Member Typedef Documentation

6.36.2.1 bases_t

```
typedef enum Timer::bases_t Timer::bases_t
```

Base de tiempo del timer

6.36.3 Member Enumeration Documentation

6.36.3.1 bases_t

```
enum Timer::bases_t
```

Base de tiempo del timer

6.36.4 Constructor & Destructor Documentation

6.36.4.1 Timer() [1/2]

```
Timer::Timer ()
```

Constructor de clase timer.
Crea un timer por defecto

6.36.4.2 Timer() [2/2]

```
Timer::Timer (
            const bases_t base,
            const Timer_Handler handler = nullptr)
```

Constructor de clase timer.
Crea un timer con los parámetros correspondientes.

Parameters

in	*handler*	Funcion a ejecutar en caso de expirar.
in	*base*	Base de tiempo del timer.

6.36.5 Member Function Documentation

6.36.5.1 GetmrStandBy()

```
bool Timer::GetmrStandBy (
            void )
```

Informa si el timer está detenido o no.

Returns

True si el timer esta detenido. False si no.

6.36.5.2 GetTimer()

```
uint32_t Timer::GetTimer (
            void ) const
```

Toma el valor al vuelo del timer en cuestion.
Lee el timer al vuelo.

Returns

uint32_t: valor del timer.

6.36.5.3 GetTmrEvent()

```
bool Timer::GetTmrEvent (
            void )
```

Indica si el timer venció o no.

Returns

True si venció. False si no.

6.36.5.4 GetTmrRun()

```
uint32_t Timer::GetTmrRun (
            void )
```

Obtiene el valor del timer.

Returns

Valor del timer.

6.36.5.5 operator bool()

```
Timer::operator bool ()  [explicit]
```

Sobrecarga de del operador de contenido.

Returns

true por timer vencido y false por no vencido.

6.36.5.6 operator"!()

```
bool Timer::operator!  ()
```

Sobrecarga de del operador de negacion.

Returns

true por timer no vencido y false por vencido.

6.36.5.7 operator=()

```
Timer & Timer::operator= (
            uint32_t time)
```

Sobrecarga de del operador de asignacion.

Parameters

in	*time*	Valor a asignar a la variable de teporizacion.

Returns

una referencia al propio objeto.

6.36.5.8 operator==()

```
bool Timer::operator== (
            uint32_t ev)
```

Sobrecarga de del operador de comparacion.
compara un valor numerico contra el flag de finalizacion del timer.

Parameters

in	*ev*	valor de comparacion (para verificar si vencio el timer).

Returns

bool: true por coincidencia, false por no coincidencia.

6.36.5.9 SetmrStandBy()

```
void Timer::SetmrStandBy (
            uint8_t accion)
```

Detiene el timer sin apagarlo.

Parameters

in	*accion*	si lo detiene o no.

6.36.5.10 SetTimer()

```
void Timer::SetTimer (
            uint32_t time)
```

Inicia un timer.
Reinicia el timer con el valor t (no lo resetea).

Parameters

in	*time*	time Tiempo del evento. Dependiente de la base de tiempos.

6.36.5.11 SetTimerBase()

```
void Timer::SetTimerBase (
            const bases_t base)
```

Setea la base de tiempo.
Fija la base de tiempo entre todos los posibles de bases_t.

Parameters

in	*base*	Base de tiempo a utilizar.

6.36.5.12 SetTmrHandler()

```
void Timer::SetTmrHandler (
            void )
```

Ejecuta el timer Handler.
Llama a la función handler entregada por el constructor (siempe que no sea nullptr).

6.36.5.13 StandByTimer()

```
void Timer::StandByTimer (
            const uint8_t accion)
```

Detiene/Arranca el timer, NO lo resetea.
lo pone o lo saca de stand-by.

Parameters

in	*accion*	RUN lo arranca, PAUSE lo pone en stand-by.

6.36.5.14 SWhandler()

```
void Timer::SWhandler (
            void ) [virtual]
```

Decremento periodico del timer. Debe ser llamada periodicamente con la base de tiempos.
Implements Callback.

6.36.5.15 TimerStart() [1/2]

```
void Timer::TimerStart (
            uint32_t time)
```

Inicia un timer.
Inicia el timer y al transcurrir el tiempo especificado se llama a la funcion apuntada por handler.

Parameters

in	*time*	Tiempo del evento. Dependiente de la base de tiempos.

6.36.5.16 TimerStart() [2/2]

```
void Timer::TimerStart (
            uint32_t time,
            const Timer_Handler handler,
            const bases_t base)
```

Inicia un timer.
Inicia el timer y al transcurrir el tiempo especificado por y time se llama a la funcion apuntada por handler.

Parameters

in	*time*	Tiempo del evento. Dependiente de la base de tiempos.
in	*handler*	Callback del evento.
in	*base*	de tiempo.

6.36.5.17 TmrEvent()

```
int8_t Timer::TmrEvent (
            void )
```

Función encargada de revisar si los timer vencieron y ejecuta automáticamente su función asignada.
Si el timer tiene una funcion asignada, debe colocarse en el while(1) para que se ejecuten automáticamente.

Returns

OKtimers si el timer venció. errorTimer si el timer no venció.

6.36.6 Friends And Related Symbol Documentation

6.36.6.1 operator==

```
bool operator== (
            uint32_t t,
            Timer & T)  [friend]
```

Sobrecarga de del operador de comparacion.
compara un valor numerico contra el flag de finalizacion del timer

Returns

bool: true por coincidencia, false por no coincidencia

6.36.7 Member Data Documentation

6.36.7.1 m_TmrBase

```
volatile bases_t Timer::m_TmrBase  [protected]
```

Base de tiempo del timer

6.36.7.2 m_TmrEvent

```
volatile bool Timer::m_TmrEvent  [protected]
```

True si el timer venció. False si el timer no venció

6.36.7.3 m_TmrHandler

```
void(* Timer::m_TmrHandler) (void)  [protected]
```

Función a ejecutar al terminar

6.36.7.4 m_TmrRun

```
volatile uint32_t Timer::m_TmrRun  [protected]
```

Tiempo de encendido del timer

6.36.7.5 m_TmrStandBy

```
volatile bool Timer::m_TmrStandBy  [protected]
```

True si el timer está detenido. False si el timer no lo está
The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/07-Timer/Timer.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/07-Timer/Timer.cpp

6.37 timers Class Reference

Clase del objeto timers El objeto timers permite agrupar todos los timers y ejecutarlos de una sola pasada. Permite ahorrar código.
`#include <Timers.h>`

Public Member Functions

- timers ()

  Constructor de clase timers.
- timers & operator<< (Timer *t)

  Sobrecarga de del operador <<.
- void TmrEvent (void)

  Funcion que revisa si los timer vencieron y los enciende.
- virtual ~timers ()=default

6.37.1 Detailed Description

Clase del objeto timers El objeto timers permite agrupar todos los timers y ejecutarlos de una sola pasada. Permite ahorrar código.

6.37.2 Constructor & Destructor Documentation

6.37.2.1 timers()

```
timers::timers ()
```

Constructor de clase timers.
Crea un timers con los parámetros correspondientes.

6.37.2.2 ~timers()

```
virtual timers::~timers ()  [virtual], [default]
```

Destructor por defecto

6.37.3 Member Function Documentation

6.37.3.1 operator<<()

```
timers & timers::operator<< (
            Timer * t)
```

Sobrecarga de del operador <<.
Agrega a la lista de timers el timer pasado.

Parameters

in	*t*	timer a agregar a la lista.

Returns

Referencia a si mismo.

6.37.3.2 TmrEvent()

```
void timers::TmrEvent (
            void )
```

Funcion que revisa si los timer vencieron y los enciende.
Esta función debe colocarse en el while(1) para que permita revisar los timers vencidos y ejecutar sus funciones asignadas. SOLO USAR SI TIENEN FUNCIONES ASIGNADAS.
The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/08-Timers/Timers.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/08-Timers/Timers.cpp

6.38 Uart Class Reference

Clase del objeto uart El objeto uart genera una comunicación asincrónica de tipo UART.
`#include <Uart.h>`
Inheritance diagram for Uart:
Collaboration diagram for Uart:

Public Types

- enum paridad_t { **NoParidad** , **par** = 2 , **impar** }
- enum bits_de_datos { **siete_bits** , **ocho_bits** }

Public Member Functions

- Uart (Pin::port_t portTx, uint8_t pinTx, Pin::port_t portRx, uint8_t pinRx, USART_Type *usart, uint32_t baudrate, bits_de_datos BitsDeDatos, paridad_t paridad, uint32_t maxRx, uint32_t maxTx)

  Constructor de la clase uart.
- void Transmit (const char *msg) override

  Transmite el mensaje indicado.
- void Transmit (const void *msg, uint32_t n) override

  Transmite el mensaje indicado.
- void * Message (void *msg, uint32_t n) override

  Devuelve el mensaje recibido.
- void SetBaudRate (uint32_t baudrate)

  Modifica el baudrate al vuelo.

Public Member Functions inherited from ComunicacionAsincronica

- ComunicacionAsincronica ()=default
- virtual ~ComunicacionAsincronica ()=default

6.38.1 Detailed Description

Clase del objeto uart El objeto uart genera una comunicación asincrónica de tipo UART.

6.38.2 Member Enumeration Documentation

6.38.2.1 bits_de_datos

`enum Uart::bits_de_datos`
Cantidad de bits de la UART

6.38.2.2 paridad_t

`enum Uart::paridad_t`
Tipo de paridad de la UART

6.38.3 Constructor & Destructor Documentation

6.38.3.1 Uart()

```
Uart::Uart (
            Pin::port_t portTx,
            uint8_t pinTx,
            Pin::port_t portRx,
            uint8_t pinRx,
            USART_Type * usart,
            uint32_t baudrate,
            bits_de_datos BitsDeDatos,
            paridad_t paridad,
            uint32_t maxRx,
            uint32_t maxTx)
```

Constructor de la clase uart.
Genera un uart con los parametros indicados

Parameters

in	*portTx*	Puerto de transmision
in	*pinTx*	bit de transmision
in	*portRx*	Puerto de recepcion

Parameters

in	*pinRx*	bit de recepcion
in	*usart*	Registro/usart a utilizar
in	*baudrate*	Baudios de la comunicacion
in	*BitsDeDatos*	Cantidad de bits por dato
in	*paridad*	Tipo de paridad de la comunicacion
in	*maxRx*	Tamaño del buffer de recepcion
in	*maxTx*	Tamaño del buffer de transmision

6.38.4 Member Function Documentation

6.38.4.1 Message()

```
void * Uart::Message (
            void * msg,
            uint32_t n)  [override], [virtual]
```

Devuelve el mensaje recibido.
Lee del buffer de recepcion n caracteres y los guarda en el mensaje.

Parameters

in	*msg*	puntero donde devolverá el dato.
in	*n*	Cantidad de caracteres a leer.

Returns

Mensaje de error. nullptr = no hay nada para leer.

Implements ComunicacionAsincronica.

6.38.4.2 SetBaudRate()

```
void Uart::SetBaudRate (
            uint32_t baudrate)
```

Modifica el baudrate al vuelo.
Cambia la velocidad de transmisión de una uart ya funcional. Verificar que se haya enviado toda la información antes de realizar este proceso

Parameters

in	*baudrate*	Bauldrate a utilizar

6.38.4.3 Transmit() [1/2]

```
void Uart::Transmit (
            const char * msg)  [override], [virtual]
```

Transmite el mensaje indicado.
Coloca el mensaje indicado en el buffer. Importante, el mensaje debe terminar en \0 (String).

Parameters

in	*msg*	Mensaje a transmitir.

Implements ComunicacionAsincronica.

6.38.4.4 Transmit() [2/2]

```
void Uart::Transmit (
            const void * msg,
            uint32_t n)  [override], [virtual]
```

Transmite el mensaje indicado.
Coloca n caracteres del mensaje indicado en el buffer.

Parameters

in	*msg*	Mensaje a transmitir.
in	*n*	Cantidad de caracteres a enviar.

Implements ComunicacionAsincronica.
The documentation for this class was generated from the following files:

- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/13-UART/Uart.h
- E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/13-UART/Uart.cpp

Chapter 7

File Documentation

7.1 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Drivers.h File Reference

Archivo Master con todos los includes de perifericos del Kit.

```
#include <Hardware/01-Teclado/teclado.h>
#include <Hardware/02-Displays/7Segmentos/C-Display/Display7Segmentos.h>
#include <Hardware/02-Displays/LCD/LCD.h>
#include <Hardware/03-Puente_H/L298N.h>
#include <Hardware/04-Reloj/Reloj.h>
#include <Hardware/05-PWM/Pwm.h>
#include <Hardware/06-PWM_Reader/PWMReader.h>
#include <Hardware/07-hc-sr04/HCSR04.h>
#include <Hardware/08-WiFi/ESP8266.h>
#include <Perifericos/04-Input/Input.h>
#include <Perifericos/05-Output/Output.h>
#include <Perifericos/07-Timer/Timer.h>
#include <Perifericos/08-Timers/Timers.h>
#include <Perifericos/11-DAC/DAC.h>
#include <Perifericos/12-ADC/ADC.h>
#include <Perifericos/13-UART/Uart.h>
#include <Perifericos/14-ComSincronica/I2C/I2CMaster.h>
```

Include dependency graph for Drivers.h:

7.1.1 Detailed Description

Archivo Master con todos los includes de perifericos del Kit.

Date

29 dic. 2024

Author

Técnico. Martinez Agustin

7.2 Drivers.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009 #ifndef DRIVERS_H_
00010 #define DRIVERS_H_
00011
00012 // Lista de todos los include de Hardware creados. Comentar los que no serán utilizados.
00013 #include <Hardware/01-Teclado/teclado.h>
00014 #include <Hardware/02-Displays/7Segmentos/C-Display/Display7Segmentos.h>
```

```
00015 #include <Hardware/02-Displays/LCD/LCD.h>
00016 #include <Hardware/03-Puente_H/L298N.h>
00017 #include <Hardware/04-Reloj/Reloj.h>
00018 #include <Hardware/05-PWM/Pwm.h>
00019 #include <Hardware/06-PWM_Reader/PWMReader.h>
00020 #include <Hardware/07-hc-sr04/HCSR04.h>
00021 #include <Hardware/08-WiFi/ESP8266.h>
00022 #include <Perifericos/04-Input/Input.h>
00023 #include <Perifericos/05-Output/Output.h>
00024 #include <Perifericos/07-Timer/Timer.h>
00025 #include <Perifericos/08-Timers/Timers.h>
00026 #include <Perifericos/11-DAC/DAC.h>
00027 #include <Perifericos/12-ADC/ADC.h>
00028 #include <Perifericos/13-UART/Uart.h>
00029 #include <Perifericos/14-ComSincronica/I2C/I2CMaster.h>
00030
00031
00032 #endif /* DRIVERS_H_ */
```

7.3 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/01-Teclado/teclado.cpp File Reference

Descripcion del modulo.
`#include <Drivers/Hardware/01-Teclado/teclado.h>`
Include dependency graph for teclado.cpp:

7.3.1 Detailed Description

Descripcion del modulo.

Date

24 jul. 2022

Author

Ing. Marcelo Trujillo

7.4 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/01-Teclado/teclado.h File Reference

Breve descripción del objetivo del Módulo.
`#include <Perifericos/02-Gpio/gpio.h>`
`#include <Perifericos/06-Callback/Callback.h>`
`#include <vector>`
Include dependency graph for teclado.h: This graph shows which files directly or indirectly include this file:

Classes

- class teclado

  Clase del objeto teclado FUNCIONAMIENTO: Este objeto permite controlar teclados matriciales cableados, eliminando el rebote mecánico. Teclado de tipo mono-usuario, mono-dedo con opción de mantener presionado una tecla.

Macros

- #define NO_KEY 0xff

7.4.1 Detailed Description

Breve descripción del objetivo del Módulo.

Date

24 jul. 2022

Author

Ing. Marcelo Trujillo

7.4.2 Macro Definition Documentation

7.4.2.1 NO_KEY

```
#define NO_KEY 0xff
```

Número representativo a "ninguna tecla se a presionado"

7.5 teclado.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013
00014 #ifndef TECLADO_H_
00015 #define TECLADO_H_
00019
      /*****************************************************************************************************************************
00020  *** INCLUDES GLOBALES
00021
      *****************************************************************************************************************************
00022 #include <Perifericos/02-Gpio/gpio.h>
00023 #include <Perifericos/06-Callback/Callback.h>
00024 #include <vector>
00025
00026
      /*****************************************************************************************************************************
00027  *** DEFINES GLOBALES
00028
      *****************************************************************************************************************************
00029
00030
      /*****************************************************************************************************************************
00031  *** MACROS GLOBALES
00032
      *****************************************************************************************************************************
00033
00034
      /*****************************************************************************************************************************
00035  *** TIPO DE DATOS GLOBALES
00036
      *****************************************************************************************************************************
00037
00038
      /*****************************************************************************************************************************
00039  *** VARIABLES GLOBALES
00040
      *****************************************************************************************************************************
00041
00042
      /*****************************************************************************************************************************
00043  *** IMPLANTACION DE UNA CLASE
00044
      *****************************************************************************************************************************
00052 class teclado : public Callback
00053 {
00054     public:
00056         #define NO_KEY      0xff
00057     private:
00058         const vector <gpio *> &m_scn;
00059         const vector <gpio *> &m_ret;
00060
00061         uint8_t         m_TeclaEstadoInicial ;
00062         uint32_t        m_TeclaCantidadDeRebotes ;
00063         uint8_t         m_BufferTeclado ;
00064         const uint8_t   m_MaxRebotes;
```

```
00065         const uint32_t  m_RebotesHold ;
00066
00067     public:
00068         teclado ( vector <gpio *> &s , vector <gpio *> &r);
00069         void SWhandler ( void );
00070         void Inicializar ( void );
00071         uint8_t Get( void );
00072         virtual ~teclado();
00073
00074     private:
00075         uint8_t TecladoHW ( void );
00076         void TecladoSW ( uint8_t TeclaEstadoActual );
00077 };
00078
00079 #endif /* TECLADO_H_ */
```

7.6 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/02-Displays/7Segmentos/A-Barrido/Barrido.h File Reference

Clase abstracta de manejo barridos de datos.
This graph shows which files directly or indirectly include this file:

Classes

- class barrido

  Clase del objeto barrido Clase abstracta pura para la generación de barridos.

7.6.1 Detailed Description

Clase abstracta de manejo barridos de datos.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.7 Barrido.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************
00009
00010
      /*****************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************
00013 #ifndef BARRIDO_H_
00014 #define BARRIDO_H_
00018
      /*****************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************
00021
00022
      /*****************************************************************************************************************
00023  *** DEFINES GLOBALES
00024
      *****************************************************************************************************************
00025
00026
      /*****************************************************************************************************************
00027  *** MACROS GLOBALES
00028
      *****************************************************************************************************************
```

```
00029
00030
      /*****************************************************************************************************************************************
00031  *** TIPO DE DATOS GLOBALES
00032
      *****************************************************************************************************************************************
00033
00034
      /*****************************************************************************************************************************************
00035  *** VARIABLES GLOBALES
00036
      *****************************************************************************************************************************************
00037
      /*****************************************************************************************************************************************
00038  *** IMPLANTACION DE UNA CLASE
00039
      *****************************************************************************************************************************************
00045 class barrido
00046 {
00047     public:
00048         barrido(){};
00049         virtual void SetDigito ( void ) = 0;
00050         virtual void Inicializar ( void ) = 0;
00051         virtual ~barrido(){};
00052 };
00053
00054 #endif /* BARRIDO_H_ */
```

7.8 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/A-Barrido/I4017.cpp File Reference

Objeto de control del integrado I4017.

`#include <Hardware/02-Displays/7Segmentos/A-Barrido/I4017.h>`

Include dependency graph for I4017.cpp:

7.8.1 Detailed Description

Objeto de control del integrado I4017.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.9 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/A-Barrido/I4017.h File Reference

Objeto de control del integrado I4017.

```
#include <Hardware/02-Displays/7Segmentos/A-Barrido/Barrido.h>
#include <Perifericos/02-Gpio/gpio.h>
#include <vector>
```

Include dependency graph for I4017.h: This graph shows which files directly or indirectly include this file:

Classes

- class I4017

  Clase del objeto I4017 El objeto I4017 permite el control del integrado del mismo nombre. Habitualmente utilizado para barrer información a tavés de sus patas.

7.9.1 Detailed Description

Objeto de control del integrado I4017.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.10 I4017.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef I4017_H_
00014 #define I4017_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include <Hardware/02-Displays/7Segmentos/A-Barrido/Barrido.h>
00022 #include <Perifericos/02-Gpio/gpio.h>
00023 #include <vector>
00024 using namespace std;
00025
00026
      /*****************************************************************************************************************************
00027  *** DEFINES GLOBALES
00028
      *****************************************************************************************************************************
00029
00030
      /*****************************************************************************************************************************
00031  *** MACROS GLOBALES
00032
      *****************************************************************************************************************************
00033
00034
      /*****************************************************************************************************************************
00035  *** TIPO DE DATOS GLOBALES
00036
      *****************************************************************************************************************************
00037
00038
      /*****************************************************************************************************************************
00039  *** VARIABLES GLOBALES
00040
      *****************************************************************************************************************************
00041
00042
      /*****************************************************************************************************************************
00043  *** IMPLANTACION DE UNA CLASE
00044
      *****************************************************************************************************************************
00050 class I4017 : public barrido
00051 {
00052     uint8_t m_indice ;
00053     const vector <gpio * > &m_pins4017;
00054     const uint8_t m_maxsalidas ;
00055
00056     public:
00057     // tiene que recibir los gpio clk y reset
00058         I4017( const vector <gpio * > &pins4017 , uint8_t maxsalidas ) : m_indice (0) , m_pins4017
      (pins4017 ) , m_maxsalidas ( (maxsalidas <= 10 ) ? maxsalidas:10) {}
00059
00060         void SetDigito ( void );
00061         void SetReset( void );
00062         void SetClock( void );
00063         void Inicializar ( void );
00064         virtual ~I4017() {};
00065 };
```

```
00066
00067 #endif /* I4017_H_ */
```

7.11 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/02-Displays/7Segmentos/B-Segmentos/I4511.cpp File Reference

Objeto de control del integrado I4511.
`#include <Hardware/02-Displays/7Segmentos/B-Segmentos/I4511.h>`
Include dependency graph for I4511.cpp:

7.11.1 Detailed Description

Objeto de control del integrado I4511.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.12 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/02-Displays/7Segmentos/B-Segmentos/I4511.h File Reference

Objeto de control del integrado I4511.
`#include <Hardware/02-Displays/7Segmentos/B-Segmentos/Segmentos.h>`
`#include <Pericericos/02-Gpio/gpio.h>`
`#include <vector>`
Include dependency graph for I4511.h: This graph shows which files directly or indirectly include this file:

Classes

- class I4511

  Clase del objeto I4511 El objeto I4511 permite el control del integrado del mismo nombre. Este integrado permite el control de un display 7 segmentos mediante una comunicación binaria en formato paralelo.

7.12.1 Detailed Description

Objeto de control del integrado I4511.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.13 I4511.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************************
00009
```

```
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef I4511_H_
00014 #define I4511_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include <Hardware/02-Displays/7Segmentos/B-Segmentos/Segmentos.h>
00022 #include <Perifericos/02-Gpio/gpio.h>
00023 #include <vector>
00024
00025 using namespace std;
00026
00027
      /*****************************************************************************************************************************
00028  *** DEFINES GLOBALES
00029
      *****************************************************************************************************************************
00030
00031
      /*****************************************************************************************************************************
00032  *** MACROS GLOBALES
00033
      *****************************************************************************************************************************
00034
00035
      /*****************************************************************************************************************************
00036  *** TIPO DE DATOS GLOBALES
00037
      *****************************************************************************************************************************
00038
00039
      /*****************************************************************************************************************************
00040  *** VARIABLES GLOBALES
00041
      *****************************************************************************************************************************
00042
00043
      /*****************************************************************************************************************************
00044  *** IMPLANTACION DE UNA CLASE
00045
      *****************************************************************************************************************************
00052 class I4511 : public segmentos
00053 {
00054     const vector <gpio * > & m_bcd ;
00055
00056     public:
00057         // tiene que recibir los gpio BCD
00058         I4511( const vector <gpio * > &bcd) :  m_bcd (bcd){}
00059         void SetSegmentos ( uint16_t ) ;
00060         void Inicializar ( void );
00061         virtual ~I4511() {};
00062 };
00063
00064 #endif /* I4511_H_ */
```

7.14 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/02-Displays/7Segmentos/B-Segmentos/Segmentos.h File Reference

Clase abstracta de manejo de segmentos binarios.
`#include "tipos.h"`
Include dependency graph for Segmentos.h: This graph shows which files directly or indirectly include this file:

Classes

- class segmentos

  Clase del objeto segmentos Clase abstracta pura para la generación de segmentos.

7.14.1 Detailed Description

Clase abstracta de manejo de segmentos binarios.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.15 Segmentos.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013
00014 #ifndef SEGMENTOS_H_
00015 #define SEGMENTOS_H_
00019
      /*****************************************************************************************************************************
00020  *** INCLUDES GLOBALES
00021
      *****************************************************************************************************************************
00022 #include "tipos.h"
00023
00024
      /*****************************************************************************************************************************
00025  *** DEFINES GLOBALES
00026
      *****************************************************************************************************************************
00027
00028
      /*****************************************************************************************************************************
00029  *** MACROS GLOBALES
00030
      *****************************************************************************************************************************
00031
00032
      /*****************************************************************************************************************************
00033  *** TIPO DE DATOS GLOBALES
00034
      *****************************************************************************************************************************
00035
00036
      /*****************************************************************************************************************************
00037  *** VARIABLES GLOBALES
00038
      *****************************************************************************************************************************
00039
00040
      /*****************************************************************************************************************************
00041  *** IMPLANTACION DE UNA CLASE
00042
      *****************************************************************************************************************************
00048 class segmentos
00049 {
00050     public:
00051         segmentos() {};
00052         virtual void SetSegmentos ( uint16_t ) = 0;
00053         virtual void Inicializar ( void ) = 0;
00054         virtual ~segmentos() {};
00055 };
00056
00057 #endif /* SEGMENTOS_H_ */
```

7.16 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/C-Display/Digito.cpp File Reference

Objeto dígito genérico para implementaciones posteriores.
`#include <Hardware/02-Displays/7Segmentos/C-Display/Digito.h>`
Include dependency graph for Digito.cpp:

Variables

- const uint8_t Tabla_Digitos_BCD_7seg [ ] = {CERO,UNO,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,OCHO,NUEVE}

7.16.1 Detailed Description

Objeto dígito genérico para implementaciones posteriores.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

7.16.2 Variable Documentation

7.16.2.1 Tabla_Digitos_BCD_7seg

`const uint8_t Tabla_Digitos_BCD_7seg[ ] = {CERO,UNO,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,OCHO,NUEVE}`
Taba de dígitos BCD en binario

7.17 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/C-Display/Digito.h File Reference

Objeto dígito genérico para implementaciones posteriores.
`#include "tipos.h"`
Include dependency graph for Digito.h: This graph shows which files directly or indirectly include this file:

Classes

- class digito

  Clase del objeto digito El objeto digito posee todas las funcionalidades y propiedades de forma que pueda ser la representación en código de un dígito. Un ejemplo de esto sería un display de 7 segmentos.

7.17.1 Detailed Description

Objeto dígito genérico para implementaciones posteriores.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

7.18 Digito.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef DIGITO_H_
00014 #define DIGITO_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include "tipos.h"
00022
00023
      /*****************************************************************************************************************************
00024  *** DEFINES GLOBALES
00025
      *****************************************************************************************************************************
00026
00027
      /*****************************************************************************************************************************
00028  *** MACROS GLOBALES
00029
      *****************************************************************************************************************************
00030
00031
      /*****************************************************************************************************************************
00032  *** TIPO DE DATOS GLOBALES
00033
      *****************************************************************************************************************************
00034
00035
      /*****************************************************************************************************************************
00036  *** VARIABLES GLOBALES
00037
      *****************************************************************************************************************************
00038
00039
      /*****************************************************************************************************************************
00040  *** IMPLANTACION DE UNA CLASE
00041
      *****************************************************************************************************************************
00048 class digito
00049 {
00050     public:
00052         typedef enum { BCD , SEGMENTOS , ASCHII } codigo_t;
00054         typedef enum { APAGAR = 0xff, PARPADEAR = 1} modo_t;
00056         typedef enum { menos = 10 ,  a , b , c , d , e , f , g , h , n , o , p , r, t , u } SIMBOLOS;
00057
00058     private:
00059         uint16_t m_Valor;
00060         const codigo_t m_Sistema;
00061
00062     public:
00064         digito( codigo_t Sistema = BCD , uint8_t Valor = APAGAR ) : m_Valor( Valor ) , m_Sistema (
      Sistema ){};
00065
00066         bool Set(uint16_t valor);
00067         uint8_t Get( void );
00068         void Clr( void );
00069
00070         virtual ~digito() {};
00071 };
00072
00073 #endif /* DIGITO_H_ */
```

7.19 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/C-Display/Display7Segmentos.cpp File Reference

Clase para la creación de displays de 7 segmentos.

```
#include <Hardware/02-Displays/7Segmentos/C-Display/Display7Segmentos.h>
```
Include dependency graph for Display7Segmentos.cpp:

7.19.1 Detailed Description

Clase para la creación de displays de 7 segmentos.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

7.20 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/02-Displays/7Segmentos/C-Display/Display7Segmentos.h File Reference

Clase para la creación de displays de 7 segmentos.

```
#include <Hardware/02-Displays/7Segmentos/A-Barrido/Barrido.h>
#include <Hardware/02-Displays/7Segmentos/B-Segmentos/Segmentos.h>
#include <Hardware/02-Displays/7Segmentos/C-Display/Digito.h>
#include <Hardware/02-Displays/7Segmentos/C-Display/GrupoDeDigitos.h>
#include <Hardware/02-Displays/Display.h>
#include <Perifericos/02-Gpio/gpio.h>
#include <Perifericos/06-Callback/Callback.h>
#include <vector>
```
Include dependency graph for Display7Segmentos.h: This graph shows which files directly or indirectly include this file:

Classes

- class display7Segmentos

  Clase del objeto display7Segmentos El objeto display7Segmentos permite el control de un display con dígitos de 7 segmentos agrupados y controlados con un integrado de barrido. Para su funcionamiento, utiliza el systick y escribe de un led a la vez a altas velocidades. La velocidad de escritura depende de la frecuencia del systick y del valor asignado a m_ticks. Para ver mejores resultados modificar dicho valor.

Macros

- #define UPDATE_TICKS (3)

7.20.1 Detailed Description

Clase para la creación de displays de 7 segmentos.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

7.20.2 Macro Definition Documentation

7.20.2.1 UPDATE_TICKS

```
#define UPDATE_TICKS (3)
```
Velocidad de escritura de los leds. MODIFICAR EN CASO DE SER NECESARIO

7.21 Display7Segmentos.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013
00014 #ifndef DISPLAY7SEGMENTOS_H_
00015 #define DISPLAY7SEGMENTOS_H_
00019
      /*****************************************************************************************************************************
00020  *** INCLUDES GLOBALES
00021
      *****************************************************************************************************************************
00022 #include <Hardware/02-Displays/7Segmentos/A-Barrido/Barrido.h>
00023 #include <Hardware/02-Displays/7Segmentos/B-Segmentos/Segmentos.h>
00024 #include <Hardware/02-Displays/7Segmentos/C-Display/Digito.h>
00025 #include <Hardware/02-Displays/7Segmentos/C-Display/GrupoDeDigitos.h>
00026 #include <Hardware/02-Displays/Display.h>
00027
00028 #include <Perifericos/02-Gpio/gpio.h>
00029 #include <Perifericos/06-Callback/Callback.h>
00030
00031 #include <vector>
00032
00033
      /*****************************************************************************************************************************
00034  *** DEFINES GLOBALES
00035
      *****************************************************************************************************************************
00036
00037
      /*****************************************************************************************************************************
00038  *** MACROS GLOBALES
00039
      *****************************************************************************************************************************
00040
00041
      /*****************************************************************************************************************************
00042  *** TIPO DE DATOS GLOBALES
00043
      *****************************************************************************************************************************
00044
00045
      /*****************************************************************************************************************************
00046  *** VARIABLES GLOBALES
00047
      *****************************************************************************************************************************
00048
00049
00050
      /*****************************************************************************************************************************
00051  *** IMPLANTACION DE UNA CLASE
00052
      *****************************************************************************************************************************
00053
00054
00055 // Esta fncion posee:
00056 // un barrido,
00057 // un segmentos,
00058 // 6 digitos,
00059 // configuracion,
00060 // heredar swhandler.
00068 class display7Segmentos : public Display, Callback
00069 {
00070     private:
00072         #define UPDATE_TICKS (3)
00073     private:
00074         vector < gruposdedigitos* > m_grupos;
00075         segmentos                  *m_seg ;
00076         barrido                   *m_dig ;
00077         uint8_t                    m_maxdigitos;
00078         uint8_t                    m_inx;
00079         uint8_t                    m_ticks;
00080
00081         vector <digito *>          m_bufferdisplay ;
00082         const uint8_t*             m_PosicionRelativa;
00083         const digito::codigo_t     m_sistema;
00084
00085     public:
00086                 display7Segmentos( vector <gruposdedigitos * > g ,
```

```
00087                                segmentos * s , barrido * b ,
00088                                const uint8_t *PosicionRelativa ,
00089                                const digito::codigo_t  sistema);
00090         void    SWhandler ( void );
00091         void    Set( uint32_t valor , uint8_t dsp );
00092         void    Write ( const uint32_t n );
00093         void    Clear ( void );
00094
00095         virtual ~display7Segmentos();
00096 };
00097
00098 #endif /* DISPLAY7SEGMENTOS_H_ */
```

7.22 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/7Segmentos/C-Display/GrupoDeDigitos.h File Reference

Clase para la agrupación de dígitos.

```
#include <Hardware/02-Displays/7Segmentos/C-Display/Digito.h>
#include "tipos.h"
```

Include dependency graph for GrupoDeDigitos.h: This graph shows which files directly or indirectly include this file:

Classes

- struct gruposdedigitos

  Estructura de grupo de dígitos.

7.22.1 Detailed Description

Clase para la agrupación de dígitos.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.23 GrupoDeDigitos.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************
00009
00010
      /*****************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************
00013 #ifndef GRUPOSDEDIGITOS_H_
00014 #define GRUPOSDEDIGITOS_H_
00018
      /*****************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************
00021 #include <Hardware/02-Displays/7Segmentos/C-Display/Digito.h>
00022 #include "tipos.h"
00023
00024
      /*****************************************************************************************************************
00025  *** DEFINES GLOBALES
00026
      *****************************************************************************************************************
00027
00028
      /*****************************************************************************************************************
00029  *** MACROS GLOBALES
```

```
00030
      ***************************************************************************************************************************************
00031
00032
      /**************************************************************************************************************************************
00033  *** TIPO DE DATOS GLOBALES
00034
      ***************************************************************************************************************************************
00035
00036
      /**************************************************************************************************************************************
00037  *** VARIABLES GLOBALES
00038
      ***************************************************************************************************************************************
00039
00040
      /**************************************************************************************************************************************
00041  *** IMPLANTACION DE UNA CLASE
00042
      ***************************************************************************************************************************************
00047 struct gruposdedigitos
00048 {
00049     const uint8_t           m_comienzo;
00050     const uint8_t           m_cantidad;
00051
00052
00053     public:
00054         gruposdedigitos(uint8_t  comienzo , uint8_t  cantidad ) :
00055         m_comienzo(comienzo) , m_cantidad (cantidad) {}
00056
00057         virtual ~gruposdedigitos() {};
00058 };
00059
00060 #endif /* GRUPOSDEDIGITOS_H_ */
```

7.24 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/02-Displays/Display.h File Reference

Clase base para objetos del tipo pantallas/displays.
`#include "tipos.h"`
Include dependency graph for Display.h: This graph shows which files directly or indirectly include this file:

Classes

- class Display

  Clase del objeto display Clase abstracta pura para la generación de displays.

7.24.1 Detailed Description

Clase base para objetos del tipo pantallas/displays.

Date

22 jun. 2022

Author

Técnico. Martinez Agustin

7.25 Display.h

Go to the documentation of this file.

```
00001
      /**************************************************************************************************************************************
00009
      /**************************************************************************************************************************************
00010  *** MODULO
00011
      ***************************************************************************************************************************************
00012 #ifndef DISPLAY_H_
00013 #define DISPLAY_H_
```

```
00017
      /*****************************************************************************************************************************************
00018  *** INCLUDES GLOBALES
00019
      *****************************************************************************************************************************************
00020 #include "tipos.h"
00021
00022
      /*****************************************************************************************************************************************
00023  *** IMPLANTACION DE LA CLASE
00024
      *****************************************************************************************************************************************
00030 class Display
00031 {
00032     public:
00033                         Display() = default;
00034         virtual void    Write ( const uint32_t n ) = 0;
00035         virtual void    Clear ( void ) = 0;
00036         virtual         ~Display() = default;
00037 };
00038
00039 #endif /* DISPLAY_H_ */
```

7.26 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/02-Displays/LCD/LCD.h File Reference

Clase para un LCD con comunicación de 4 patas, solo escritura.

```
#include <Hardware/02-Displays/Display.h>
#include <Perifericos/02-Gpio/gpio.h>
#include <Perifericos/06-Callback/Callback.h>
#include <vector>
```

Include dependency graph for LCD.h: This graph shows which files directly or indirectly include this file:

Classes

- class LCD

  Clase del objeto lcd El objeto lcd permite el manejo de displays digitales mediante comunicación de 4 bits.

Macros

- #define CLEAR_DISPLAY 0b00000001
- #define RETURN_HOME 0b00000010
- #define ENTRY_MODE_SET 0b00000100 /** 1 I/D S*/
- #define DISPLAY_CONTROL 0b00001000 /** 1 D C B*/
- #define CURSOR_DISPLAY_SHIFT 0b00010000 /** 1 S/C R/L - -*/
- #define FUNCTION_SET 0b00100000 /** 1 DL N F - -*/
- #define SET_CGRAM 0b01000000 /** 1 ACG ACG ACG ACG ACG ACG*/
- #define SET_DDRAM 0b10000000 /** 1 ADD ADD ADD ADD ADD ADD ADD*/

7.26.1 Detailed Description

Clase para un LCD con comunicación de 4 patas, solo escritura.

Date

22 jun. 2022

Author

Técnico. Martinez Agustin

7.26.2 Macro Definition Documentation

7.26.2.1 CLEAR_DISPLAY

```
#define CLEAR_DISPLAY 0b00000001
```

Instruccion de limpieza del display

7.26.2.2 CURSOR_DISPLAY_SHIFT

```
#define CURSOR_DISPLAY_SHIFT 0b00010000 /** 1 S/C R/L - -*/
```

Instruccion de cursor del display

7.26.2.3 DISPLAY_CONTROL

```
#define DISPLAY_CONTROL 0b00001000 /** 1 D C B*/
```

Instruccion de control del display

7.26.2.4 ENTRY_MODE_SET

```
#define ENTRY_MODE_SET 0b00000100 /** 1 I/D S*/
```

Instruccion de entrada de escritura del display

7.26.2.5 FUNCTION_SET

```
#define FUNCTION_SET 0b00100000 /** 1 DL N F - -*/
```

Instruccion de modo de funcionamiento del display

7.26.2.6 RETURN_HOME

```
#define RETURN_HOME 0b00000010
```

Instruccion de regreso a posición 0 del display

7.26.2.7 SET_CGRAM

```
#define SET_CGRAM 0b01000000 /** 1 ACG ACG ACG ACG ACG ACG*/
```

Instruccion de seteo de CGRAM del display

7.26.2.8 SET_DDRAM

```
#define SET_DDRAM 0b10000000 /** 1 ADD ADD ADD ADD ADD ADD ADD*/
```

Instruccion de seteo de DDRAM del display

7.27 LCD.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef LCD_H_
00014 #define LCD_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include <Hardware/02-Displays/Display.h>
00022 #include <Perifericos/02-Gpio/gpio.h>
00023 #include <Perifericos/06-Callback/Callback.h>
00024 #include <vector>
00025
      /*****************************************************************************************************************************
00026  *** DEFINES GLOBALES
00027
      *****************************************************************************************************************************
00028
00029
      /*****************************************************************************************************************************
00030  *** MACROS GLOBALES
00031
      *****************************************************************************************************************************
00032
```

```
00033
      /*****************************************************************************************************************************
00034  *** TIPO DE DATOS GLOBALES
00035
      *****************************************************************************************************************************
00036
00037
      /*****************************************************************************************************************************
00038  *** VARIABLES GLOBALES
00039
      *****************************************************************************************************************************
00040
00041
      /*****************************************************************************************************************************
00042  *** IMPLANTACION DE LA CLASE
00043
      *****************************************************************************************************************************
00049 class LCD : public Display, Callback
00050 {
00051     public:
00053         enum { d7 = 0 ,  d6 , d5 , d4 , rs , enable };
00054
00055     private:
00057         #define CLEAR_DISPLAY           0b00000001
00059         #define RETURN_HOME             0b00000010
00061         #define ENTRY_MODE_SET          0b00000100
00063         #define DISPLAY_CONTROL         0b00001000
00065         #define CURSOR_DISPLAY_SHIFT    0b00010000
00067         #define FUNCTION_SET            0b00100000
00069         #define SET_CGRAM               0b01000000
00071         #define SET_DDRAM               0b10000000
00072
00073     private:
00074         const vector<gpio*> m_salidas;
00075         uint8_t m_estado;
00076         uint8_t *m_buffer;
00077         uint8_t m_filas;
00078         uint8_t m_columnas;
00079         uint32_t m_delay;
00080         uint8_t m_barrido;
00081         uint8_t m_pos;
00082         uint8_t m_error;
00083
00084         enum { s_eigth_bits = 0 , s_four_bits , s_config_display , s_config_cursor , s_clear , s_print
      , s_row };
00085
00086     public:
00087         LCD( vector<gpio*> &salidas );
00088         void Inicializar( const uint8_t filas , const uint8_t columnas );
00089         void Write ( const char *s );
00090         void Write ( const int32_t n );
00091         LCD& operator= ( const char *s );
00092         void WriteAt( const int8_t *a , const uint8_t fila , const uint8_t columna );
00093         void WriteAt ( const int32_t n , const uint8_t fila , const uint8_t columna);
00094         void Clear( void );
00095         virtual ~LCD();
00096
00097     protected:
00098         void SWhandler ( void );
00099
00100     private:
00101         void WriteInstruction( const uint8_t data , const uint8_t mode );
00102         uint32_t Pow ( uint32_t base , uint32_t exp );
00103 };
00104
00105 #endif /* LCD_H_ */
```

7.28 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/03-Puente_H/L298N.h File Reference

Clase del módulo de puente H LN298N.

```
#include <Hardware/03-Puente_H/PuenteH.h>
#include <Perifericos/02-Gpio/gpio.h>
```

Include dependency graph for L298N.h: This graph shows which files directly or indirectly include this file:

Classes

- class L298N

Clase del objeto L298N El objeto L298N realiza las acciones de control de dos motores controlados por el correspondiente periférico.

7.28.1 Detailed Description

Clase del módulo de puente H LN298N.

Date

25 sep. 2022

Author

Técnico. Martinez Agustin

7.29 L298N.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010 #ifndef L298N_H_
00011 #define L298N_H_
00015
      /*****************************************************************************************************************************
00016  *** INCLUDES GLOBALES
00017
      *****************************************************************************************************************************
00018 #include <Hardware/03-Puente_H/PuenteH.h>
00019 #include <Perifericos/02-Gpio/gpio.h>
00020
      /*****************************************************************************************************************************
00021  *** DEFINES GLOBALES
00022
      *****************************************************************************************************************************
00023
00024
      /*****************************************************************************************************************************
00025  *** MACROS GLOBALES
00026
      *****************************************************************************************************************************
00027
00028
      /*****************************************************************************************************************************
00029  *** TIPO DE DATOS GLOBALES
00030
      *****************************************************************************************************************************
00031
00032
      /*****************************************************************************************************************************
00033  *** IMPLANTACION DE LA CLASE
00034
      *****************************************************************************************************************************
00040 class L298N : protected Puente_H
00041 {
00042     private:
00043         gpio*   &m_motor1_a;
00044         gpio*   &m_motor1_b;
00045         gpio*   &m_motor2_a;
00046         gpio*   &m_motor2_b;
00047     public:
00048                 L298N( gpio* &_motorDer_a , gpio* &_motorDer_b , gpio* &_motorIzq_a , gpio*
      &_motorIzq_b );
00049         void    Inicializar( void );
00050         void    GirarIzq ( void );
00051         void    GirarDer ( void );
00052         void    Girar ( const uint8_t direccion );
00053         void    Frenar ( void );
00054         void    Avanzar ( void );
00055         void    Retroceder ( void );
00056         virtual ~L298N();
00057 };
00058
00059 #endif /* L298N_H_ */
```

7.30 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/03-Puente_H/PuenteH.h File Reference

Clase base para objetos del tipo Puente H.
`#include "tipos.h"`
Include dependency graph for PuenteH.h: This graph shows which files directly or indirectly include this file:

Classes

- class Puente_H

  Clase del objeto Puente_H El objeto Puente_H es la interfaz abstracta pura de cualquier puente H que se desee realizar.

7.30.1 Detailed Description

Clase base para objetos del tipo Puente H.

Date

25 sep. 2022

Author

Técnico. Martinez Agustin

7.31 PuenteH.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010 #ifndef PUENTEH_H_
00011 #define PUENTEH_H_
00015
      /*****************************************************************************************************************************
00016  *** INCLUDES GLOBALES
00017
      *****************************************************************************************************************************
00018 #include "tipos.h"
00019
      /*****************************************************************************************************************************
00020  *** DEFINES GLOBALES
00021
      *****************************************************************************************************************************
00022
00023
      /*****************************************************************************************************************************
00024  *** MACROS GLOBALES
00025
      *****************************************************************************************************************************
00026
      /*****************************************************************************************************************************
00027  *** TIPO DE DATOS GLOBALES
00028
      *****************************************************************************************************************************
00029
00030
      /*****************************************************************************************************************************
00031  *** IMPLANTACION DE LA CLASE
00032
      *****************************************************************************************************************************
00038 class Puente_H
00039 {
00040     public:
00042         enum { IZQUIERDA = 0 , DERECHA};
00043     public:
00044                         Puente_H( ) = default;
00045         virtual void    Inicializar(void) = 0;
00046         virtual void    GirarIzq(void) = 0;
00047         virtual void    GirarDer(void) = 0;
00048         virtual void    Girar ( uint8_t direccion ) = 0;
00049         virtual void    Frenar ( void ) = 0;
00050         virtual void    Avanzar( void ) = 0;
```

```
00051         virtual void    Retroceder( void ) = 0;
00052         virtual         ~Puente_H() = default;
00053 };
00054
00055 #endif /* PUENTEH_H_ */
```

7.32 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/04-Reloj/Reloj.h File Reference

Objeto que guardará el tiempo desde que se creo.
`#include <Perifericos/06-Callback/Callback.h>`
`#include "tipos.h"`
Include dependency graph for Reloj.h: This graph shows which files directly or indirectly include this file:

Classes

- class Reloj

  Clase del objeto Reloj.

7.32.1 Detailed Description

Objeto que guardará el tiempo desde que se creo.

Date

27 nov. 2022

Author

Técnico Martinez Agustin

7.33 Reloj.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef RELOJ_H_
00014 #define RELOJ_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include <Perifericos/06-Callback/Callback.h>
00022 #include "tipos.h"
00023
      /*****************************************************************************************************************************
00024  *** DEFINES GLOBALES
00025
      *****************************************************************************************************************************
00026
00027
      /*****************************************************************************************************************************
00028  *** MACROS GLOBALES
00029
      *****************************************************************************************************************************
00030
00031
      /*****************************************************************************************************************************
00032  *** TIPO DE DATOS GLOBALES
00033
      *****************************************************************************************************************************
00034
00035
      /*****************************************************************************************************************************
```

```
00036  *** VARIABLES GLOBALES
00037
       ********************************************************************************************************************************
00038
00039
       /********************************************************************************************************************************
00040  *** IMPLANTACION DE LA CLASE
00041
       ********************************************************************************************************************************
00048 class Reloj : public Callback
00049 {
00050     private:
00051         int32_t     m_hora;
00052         int32_t     m_minutos;
00053         int32_t     m_segundos;
00054         uint32_t    m_cont;
00055
00056     public:
00057                 Reloj();
00058         int32_t GetHour( void ) const;
00059         int32_t GetMin( void ) const;
00060         int32_t GetSeg( void ) const;
00061         void    Reset ( void );
00062         void    SetTime ( const int32_t _hour , const int32_t _min = -1 , const int32_t _seg = -1);
00063
00064         virtual ~Reloj();
00065
00066     private:
00067         void    Actualizar( void );
00068
00069     protected:
00070         void    SWhandler ( void );
00071 };
00072
00073 #endif /* RELOJ_H_ */
```

7.34 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/05-PWM/Pwm.h File Reference

Generador de PWM sin interrupción.

```
#include <Drivers/Perifericos/10-SCTimer/SCtimer.h>
#include <Perifericos/01-Pin/Pin.h>
```

Include dependency graph for Pwm.h: This graph shows which files directly or indirectly include this file:

Classes

- class Pwm

  Clase del objeto Pwm.

7.34.1 Detailed Description

Generador de PWM sin interrupción.

Date

7 oct. 2022

Author

Técnico Martinez Agustin

7.35 Pwm.h

Go to the documentation of this file.

```
00001
       /********************************************************************************************************************************
00009
00010 #ifndef PWM_H_
00011 #define PWM_H_
00015
       /********************************************************************************************************************************
```

```
00016  *** INCLUDES GLOBALES
00017
       ********************************************************************************************************************************
00018 #include <Drivers/Perifericos/10-SCTimer/SCtimer.h>
00019 #include <Perifericos/01-Pin/Pin.h>
00020
       /*******************************************************************************************************************************
00021  *** DEFINES GLOBALES
00022
       ********************************************************************************************************************************
00023
00024
       /*******************************************************************************************************************************
00025  *** MACROS GLOBALES
00026
       ********************************************************************************************************************************
00027
00028
       /*******************************************************************************************************************************
00029  *** TIPO DE DATOS GLOBALES
00030
       ********************************************************************************************************************************
00031
00032
       /*******************************************************************************************************************************
00033  *** IMPLANTACION DE LA CLASE
00034
       ********************************************************************************************************************************
00045 class Pwm : protected SCtimer , protected Pin
00046 {
00047 public:
00049     typedef enum { SEG = 0, MILI_SEG , MICRO_SEG } pwm_time_unit_t;
00051     typedef enum { CHANNEL_1 = 1 , CHANNEL_2 , CHANNEL_3 , CHANNEL_4 , CHANNEL_5 , CHANNEL_6 }
      pwm_channel_t;
00053     enum activity_t       { low , high };
00054
00055 protected:
00056     const   uint8_t     m_activity;
00057             uint32_t    m_ton;
00058             uint32_t    m_toff;
00059     const   uint8_t     m_pwm_channel;
00060
00061 public:
00062             Pwm( port_t puerto , uint8_t bit , uint8_t actividad , pwm_channel_t number );
00063     void    Inicializar( uint32_t ton , uint32_t toff , pwm_time_unit_t t = MICRO_SEG );
00064     void    SetTon( uint32_t time , pwm_time_unit_t t = MICRO_SEG );
00065     void    SetPeriod( uint32_t time , pwm_time_unit_t t = MICRO_SEG );
00066     void    On();
00067     void    Off();
00068     virtual ~Pwm();
00069 };
00070
00071 #endif /* PWM_H_ */
```

7.36 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/06-PWM_Reader/PWMReader.h File Reference

Pata que lee tamaños de pulsos de entrada.
`#include <Drivers/Perifericos/09-MultiRateTimer/MRThandler.h>`
`#include <Perifericos/03-PinInterrupt/Pininterrupt.h>`
Include dependency graph for PWMReader.h: This graph shows which files directly or indirectly include this file:

Classes

- class PWM_Reader

  Clase del objeto PWM_Reader.

7.36.1 Detailed Description

Pata que lee tamaños de pulsos de entrada.

Date

22 jun. 2022

Author

Técnico. Martinez Agustin

7.37 PWMReader.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef PWM_READER_H_
00014 #define PWM_READER_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include <Drivers/Perifericos/09-MultiRateTimer/MRThandler.h>
00022 #include <Perifericos/03-PinInterrupt/Pininterrupt.h>
00023
      /*****************************************************************************************************************************
00024  *** DEFINES GLOBALES
00025
      *****************************************************************************************************************************
00026
00027
      /*****************************************************************************************************************************
00028  *** MACROS GLOBALES
00029
      *****************************************************************************************************************************
00030
00031
      /*****************************************************************************************************************************
00032  *** TIPO DE DATOS GLOBALES
00033
      *****************************************************************************************************************************
00034
00035
      /*****************************************************************************************************************************
00036  *** VARIABLES GLOBALES
00037
      *****************************************************************************************************************************
00038
00039
      /*****************************************************************************************************************************
00040  *** IMPLANTACION DE LA CLASE
00041
      *****************************************************************************************************************************
00056 class PWM_Reader : protected PinInterrupt , protected MRThandler
00057 {
00058     private:
00059         uint32_t m_pulse_on;
00060
00061     public:
00062                     PWM_Reader( port_t puerto , uint8_t bit , mode_t modo ,
00063                             activity_t activity , MRT_timer_channels timer_channel );
00064         void        Inicializar( void );
00065         uint32_t    GetPulseOn( void ) const;
00066         void        Off( void );
00067         void        On( void );
00068         virtual     ~PWM_Reader(){};
00069     protected:
00070         void        GpioHandler( void );
00071 };
00072
00073 #endif /* PWM_READER_H_ */
```

7.38 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/07-hc-sr04/distancia.h File Reference

Clase base para objetos medidores de distancias.

`#include "tipos.h"`

Include dependency graph for distancia.h: This graph shows which files directly or indirectly include this file:

Classes

- class distancia

  Clase del objeto distancia Clase abstracta pura para la generación de HCS-R04.

7.38.1 Detailed Description

Clase base para objetos medidores de distancias.

Date

22 jun. 2022

Author

Técnico. Martinez Agustin

7.39 distancia.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009 #ifndef DISTANCIA_H_
00010 #define DISTANCIA_H_
00014
      /*****************************************************************************************************************************
00015  *** INCLUDES GLOBALES
00016
      *****************************************************************************************************************************
00017 #include "tipos.h"
00018
      /*****************************************************************************************************************************
00019  *** DEFINES GLOBALES
00020
      *****************************************************************************************************************************
00021
00022
      /*****************************************************************************************************************************
00023  *** MACROS GLOBALES
00024
      *****************************************************************************************************************************
00025
00026
      /*****************************************************************************************************************************
00027  *** TIPO DE DATOS GLOBALES
00028
      *****************************************************************************************************************************
00029
00030
      /*****************************************************************************************************************************
00031  *** IMPLANTACION DE LA CLASE
00032
      *****************************************************************************************************************************
00038 class distancia
00039 {
00040 public:
00041                         distancia () = default;
00042     virtual uint32_t    GetDistancia() = 0;
00043     virtual bool        operator==( uint32_t a ) = 0;
00044     virtual             ~distancia() = default;
00045 };
00046
00047 #endif /* DISTANCIA_H_ */
```

7.40 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Hardware/07-hc-sr04/HCSR04.h File Reference

Clase del sensor ultrasónico HCSR04.

```
#include <Hardware/05-PWM/Pwm.h>
#include <Hardware/06-PWM_Reader/PWMReader.h>
#include <Hardware/07-hc-sr04/distancia.h>
```

Include dependency graph for HCSR04.h: This graph shows which files directly or indirectly include this file:

Classes

- class HC_SR04

  Clase del objeto HC_SR04 El objeto HC_SR04 Mide distancia mediante el uso de un ultrasónico. Debido a los tiempos muy pequeños de uso, no se recomienda utilizar en grandes cantidades.

Macros

- #define PERIODO 80
- #define CALC_DISTANCIA(a)
- #define DISTANCIA_MAX 400

7.40.1 Detailed Description

Clase del sensor ultrasónico HCSR04.

Date

22 jun. 2022

Author

Técnico. Martinez Agustin

7.40.2 Macro Definition Documentation

7.40.2.1 CALC_DISTANCIA

```
#define CALC_DISTANCIA(
                a)
```

Value:

```
( ((a)*164) / 10000 )
```

Fórmula de cálculo tiempo->distancia

7.40.2.2 DISTANCIA_MAX

```
#define DISTANCIA_MAX 400
```

Distancia máxima en centimetros

7.40.2.3 PERIODO

```
#define PERIODO 80
```

Perido del PWM En milisegundos

7.41 HCSR04.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************************
00009 #ifndef HCSR04_H_
00010 #define HCSR04_H_
00014
      /*****************************************************************************************************************************************
00015  *** INCLUDES GLOBALES
00016
      *****************************************************************************************************************************************
00017 #include <Hardware/05-PWM/Pwm.h>
00018 #include <Hardware/06-PWM_Reader/PWMReader.h>
00019 #include <Hardware/07-hc-sr04/distancia.h>
00020
      /*****************************************************************************************************************************************
00021  *** DEFINES GLOBALES
00022
      *****************************************************************************************************************************************
00023
```

```
00024
      /**************************************************************************************************************************************************
00025  *** MACROS GLOBALES
00026
      **************************************************************************************************************************************************/
00027
00028
      /**************************************************************************************************************************************************
00029  *** TIPO DE DATOS GLOBALES
00030
      **************************************************************************************************************************************************/
00031
00032
      /**************************************************************************************************************************************************
00033  *** IMPLANTACION DE LA CLASE
00034
      **************************************************************************************************************************************************/
00046 class HC_SR04 : protected distancia
00047 {
00048     private:
00050         #define PERIODO                 80
00052         #define CALC_DISTANCIA(a)       ( ((a)*164) / 10000 )
00053     public:
00055         #define DISTANCIA_MAX   400
00056     private:
00057         PWM_Reader*       &m_rx;
00058         Pwm*             &m_tx;
00059         uint32_t        m_distancia;
00060         bool            m_stop;
00061
00062     public:
00063                     HC_SR04( PWM_Reader* &rx , Pwm* &tx );
00064         void        Inicializar( void );
00065         uint32_t    GetDistancia( void );
00066         void        Off( void );
00067         void        On( void );
00068         bool        operator==( const uint32_t a);
00069         bool        operator<=( const uint32_t a );
00070         bool        operator>=( const uint32_t a );
00071         bool        operator<( const uint32_t a );
00072         bool        operator>( const uint32_t a );
00073
00074         virtual     ~HC_SR04(){};
00075 };
00076
00077 #endif /* HCSR04_H_ */
```

7.42 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/08-WiFi/ESP8266.cpp File Reference

Breve descripcion del modulo.
`#include <Hardware/08-WiFi/ESP8266.h>`
Include dependency graph for ESP8266.cpp:

7.42.1 Detailed Description

Breve descripcion del modulo.

Date

2 mar. 2023

Author

Técnico Martinez Agustín

Version

v1.0

7.43 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Hardware/08-WiFi/ESP8266.h File Reference

Breve descripción del Módulo.

```
#include <Drivers/Perifericos/13-UART/Uart.h>
#include <Perifericos/07-Timer/Timer.h>
```

Include dependency graph for ESP8266.h: This graph shows which files directly or indirectly include this file:

Classes

- class ESP8266

  Clase del objeto ESP8266 El objeto ESP8266 permite la simple utilización del módulo arduino ESP8266 y el ESP01 mediante comandos AT. El módulo debe estar por defecto en la velocidad DEFAULT_ESP01_BAUDRATE. El módulo será conectado como cliente en modo TCP/UDP y con transmisión libre, sin filtros. La data llega y se envía cruda (como está). Por falta de material la clase no fue probada por completo. Sí se probó la inicialización y conección a internet, no se probó la conección a un servidor. Todas sus funciones son bloqueantes o poseen un timeout, debe ser tenido en cuenta a la hora de utilizar este driver.

Macros

- #define DEFAULT_ESP01_BAUDRATE 115200
- #define SEG_ESP01_TIMEOUT 20

7.43.1 Detailed Description

Breve descripción del Módulo.

Date

2 mar. 2023

Author

Técnico Martinez Agustín

Version

v1.0

7.43.2 Macro Definition Documentation

7.43.2.1 DEFAULT_ESP01_BAUDRATE

```
#define DEFAULT_ESP01_BAUDRATE 115200
```

Velocidad de transmision del ESP por defecto, antes de haber sido modificado

7.43.2.2 SEG_ESP01_TIMEOUT

```
#define SEG_ESP01_TIMEOUT 20
```

Tiempo por default de espera para la realización de los comandos AT

7.44 ESP8266.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00010
00011
      /*****************************************************************************************************************************
00012  *** MODULO
00013
      *****************************************************************************************************************************
00014 #ifndef ESP8266_H_
00015 #define ESP8266_H_
00016
00017
      /*****************************************************************************************************************************
00018  *** INCLUDES GLOBALES
```

```
00019
      /*****************************************************************************************************************************
00020 #include <Drivers/Perifericos/13-UART/Uart.h>
00021 #include <Perifericos/07-Timer/Timer.h>
00022
      /*****************************************************************************************************************************
00023  *** DEFINES GLOBALES
00024
      *****************************************************************************************************************************
00025
00026
      /*****************************************************************************************************************************
00027  *** MACROS GLOBALES
00028
      *****************************************************************************************************************************
00029
00030
      /*****************************************************************************************************************************
00031  *** TIPO DE DATOS GLOBALES
00032
      *****************************************************************************************************************************
00033
00034
      /*****************************************************************************************************************************
00035  *** VARIABLES GLOBALES
00036
      *****************************************************************************************************************************
00037
00038
      /*****************************************************************************************************************************
00039  *** IMPLANTACION DE LA CLASE
00040
      *****************************************************************************************************************************
00050 class ESP8266 : protected Uart
00051 {
00052 public:
00054     typedef enum { TCP = 0 , UDP = 1 } conection_type;
00056     typedef enum { ERROR = 0 , NOTHING , INITIALIZED , CONNECT_TO_WIFI , CONNECT_TO_SERVER }
      status_type;
00057 private:
00059     #define DEFAULT_ESP01_BAUDRATE      115200
00061     #define SEG_ESP01_TIMEOUT           20
00062 private:
00063     const uint32_t  m_baudrate;
00064     int8_t *        m_address;
00065     int8_t *        m_password;
00066     int8_t *        m_IP;
00067     status_type      m_status;
00068
00069     uint8_t         m_aux;
00070
00071 public:
00072     ESP8266( Pin::port_t _portTx , uint8_t _pinTx , Pin::port_t _portRx , uint8_t _pinRx , USART_Type
      * usart , uint32_t baudrate );
00073     void Inicializar( void );
00074     status_type ConnectToWifi ( const int8_t * wifi_address , const int8_t * wifi_password , uint32_t
      seg_timeout = SEG_ESP01_TIMEOUT );
00075     void DisconnectToWifi ( void );
00076     void SetIP ( int8_t *ip );
00077     int8_t* GetIP( void ) const;
00078     bool ConnectToServer ( conection_type _mode , const int8_t* server_ip , const int8_t* server_port
      , uint32_t seg_timeout = SEG_ESP01_TIMEOUT );
00079     void DisconnectToServer ( void );
00080     void Transmit ( const char * msg) ;
00081     void Transmit ( const void * msg , uint32_t n );
00082     void* Message ( void * msg , uint32_t n );
00083     status_type GetStatus ( void ) const;
00084     bool IsConnectedToWifi( void ) const;
00085     bool IsConnectedToServer( void ) const;
00086     virtual ~ESP8266();
00087
00088 private:
00089     bool LeerOk ( void );
00090     int8_t* toString ( const uint32_t n );
00091     uint32_t Strlen ( const int8_t * a );
00092     uint32_t pow ( uint32_t num , uint32_t exp );
00093 };
00094
00095 #endif /* ESP8266_H_ */
```

7.45 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/01-Pin/Pin.h File Reference

Clase Abstracta de cualquier pin del microcontrolador.
`#include "tipos.h"`
Include dependency graph for Pin.h: This graph shows which files directly or indirectly include this file:

Classes

- class Pin

  Clase del objeto Pin.

Variables

- const uint8_t IOCON_INDEX_PIO0 [ ] = { 17,11,6,5,4,3,16,15,4,13,8,7,2,1,18,10,9,0,30,29,28,27,26,25,24,23,22,21,20,0,0,35}
- const uint8_t IOCON_INDEX_PIO1 [ ] = { 36,37,3,41,42,43,46,49,31,32,55,54,33,34,39,40,44,45,47,48,52,53,0,0,0,0,0,0,0,50,5

7.45.1 Detailed Description

Clase Abstracta de cualquier pin del microcontrolador.

Date

10 ene. 2023

Author

Técnico Martinez Agustin

7.46 Pin.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************
00009
00010
      /*****************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************
00013 #ifndef PIN_H_
00014 #define PIN_H_
00018
00019
      /*****************************************************************************************************************
00020  *** INCLUDES GLOBALES
00021
      *****************************************************************************************************************
00022 #include "tipos.h"
00023
00024
      /*****************************************************************************************************************
00025  *** DEFINES GLOBALES
00026
      *****************************************************************************************************************
00027
00028
      /*****************************************************************************************************************
00029  *** MACROS GLOBALES
00030
      *****************************************************************************************************************
00031
00032
      /*****************************************************************************************************************
00033  *** TIPO DE DATOS GLOBALES
00034
      *****************************************************************************************************************
00035
00036
      /*****************************************************************************************************************
00037  *** VARIABLES GLOBALES
```

```
00038
      ***************************************************************************************************************************************
00040 const uint8_t IOCON_INDEX_PIO0[] = {
      17,11,6,5,4,3,16,15,4,13,8,7,2,1,18,10,9,0,30,29,28,27,26,25,24,23,22,21,20,0,0,35};
00042 const uint8_t IOCON_INDEX_PIO1[] = {
      36,37,3,41,42,43,46,49,31,32,55,54,33,34,39,40,44,45,47,48,52,53,0,0,0,0,0,0,0,50,51};
00043
      /***************************************************************************************************************************************
00044  *** IMPLANTACION DE LA CLASE
00045
      ***************************************************************************************************************************************
00052 class Pin
00053 {
00054     public:
00056         typedef enum port_t           { port0 , port1 } port_t;
00057         enum max_bits_port_t{ b_port0 = 31 , b_port1 = 9 };
00059         typedef enum error_t      { error = 2 , ok } error_t;
00060
00061     public:
00062         const   port_t m_port ;
00063         const   uint8_t m_bit ;
00064         int8_t  m_error ;
00065
00066     public:
00067         Pin( port_t port , uint8_t bit );
00068         virtual ~Pin() {};
00069 };
00070 #endif /* PIN_H_ */
```

7.47 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/02-Gpio/gpio.cpp File Reference

Descripcion del modulo.
`#include <Perifericos/02-Gpio/gpio.h>`
Include dependency graph for gpio.cpp:

7.47.1 Detailed Description

Descripcion del modulo.
Objeto DAC.
Módulo con clase de manejo de GPIO.

Date

22 jun. 2022

Author

Ing. Marcelo Trujillo

Date

12 ene. 2023

Author

Técnico Martinez Agustin

7.48 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/02-Gpio/gpio.h File Reference

Módulo con clase de manejo de GPIO.
`#include <LPC845.h>`
`#include <Perifericos/01-Pin/Pin.h>`
`#include <Perifericos/02-Gpio/InOut.h>`
Include dependency graph for gpio.h: This graph shows which files directly or indirectly include this file:

Classes

- class gpio

  Clase del objeto gpio.

7.48.1 Detailed Description

Módulo con clase de manejo de GPIO.

Date

22 jun. 2022

Author

Ing. Marcelo Trujillo

7.49 gpio.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013
00014 #ifndef GPIO_H_
00015 #define GPIO_H_
00019
      /*****************************************************************************************************************************
00020  *** INCLUDES GLOBALES
00021
      *****************************************************************************************************************************
00022 #include <LPC845.h>
00023 #include <Perifericos/01-Pin/Pin.h>
00024 #include <Perifericos/02-Gpio/InOut.h>
00025
00026
      /*****************************************************************************************************************************
00027  *** DEFINES GLOBALES
00028
      *****************************************************************************************************************************
00029
00030
      /*****************************************************************************************************************************
00031  *** MACROS GLOBALES
00032
      *****************************************************************************************************************************
00033
00034
      /*****************************************************************************************************************************
00035  *** TIPO DE DATOS GLOBALES
00036
      *****************************************************************************************************************************
00037
00038
      /*****************************************************************************************************************************
00039  *** VARIABLES GLOBALES
00040
      *****************************************************************************************************************************
00041
00042
      /*****************************************************************************************************************************
00043  *** IMPLANTACION DE LA CLASE
00044
      *****************************************************************************************************************************
00051 class gpio : public InOut , public Pin
00052 {
00053     public:
00055         typedef enum direction_t { input , output } direction_t;
00056         enum power_t      { off , on };
00058         typedef enum mode_t    { pushpull = 0 , opencolector , inactive = 0 , pulldown , pullup ,
      repeater } mode_t;          // el orden esta atado a las hojas de datos
00060         typedef enum activity_t       { low , high } activity_t;
```

```
00061         enum interrupt_mode_t { rising_edge = 0 , falling_edge , rising_falling_edge , low_level ,
      high_level };
00062
00063     protected:
00064         const   mode_t m_mode ;
00065         direction_t m_direction ;
00066         const activity_t m_activity ;
00067
00068     public:
00069         gpio ( port_t port , uint8_t bit , mode_t mode , direction_t direction , activity_t activity =
      high );
00070
00071         uint8_t SetPin ( void ) override;
00072         uint8_t ClrPin ( void ) override;
00073         uint8_t SetTogglePin ( void ) override;
00074         uint8_t SetDir ( void ) override;
00075         uint8_t SetToggleDir ( void ) override;
00076         uint8_t GetPin ( void ) override;
00077         uint8_t SetPinMode ( void ) override;
00078         uint8_t SetPinResistor ( void ) override;
00079         gpio&   operator= ( uint8_t a );
00080
00081         virtual ~gpio() = default;
00082 };
00083
00084 #endif /* GPIO_H_ */
```

7.50 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/02-Gpio/InOut.h File Reference

Clase Abstracta Pura de las GPIO.
`#include <Perifericos/01-Pin/Pin.h>`
Include dependency graph for InOut.h: This graph shows which files directly or indirectly include this file:

Classes

- class InOut

  Clase del objeto InOut.

7.50.1 Detailed Description

Clase Abstracta Pura de las GPIO.

Date

22 jun. 2022

Author

Ing. Marcelo Trujillo

7.51 InOut.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef IN_OUT_H_
00014 #define IN_OUT_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include <Perifericos/01-Pin/Pin.h>
00022
      /*****************************************************************************************************************************
```

```
00023  *** DEFINES GLOBALES
00024
       ********************************************************************************************************************************
00025
00026
       /*******************************************************************************************************************************
00027  *** MACROS GLOBALES
00028
       ********************************************************************************************************************************
00029
00030
       /*******************************************************************************************************************************
00031  *** TIPO DE DATOS GLOBALES
00032
       ********************************************************************************************************************************
00033
00034
       /*******************************************************************************************************************************
00035  *** VARIABLES GLOBALES
00036
       ********************************************************************************************************************************
00037
00038
       /*******************************************************************************************************************************
00039  *** IMPLANTACION DE LA CLASE
00040
       ********************************************************************************************************************************
00047 class InOut
00048 {
00049     public:
00050                         InOut () = default;
00051         virtual uint8_t SetPin ( void )       = 0;
00052         virtual uint8_t ClrPin ( void )       = 0;
00053         virtual uint8_t SetTogglePin ( void )   = 0;
00054         virtual uint8_t SetDir ( void )       = 0;
00055         virtual uint8_t SetToggleDir ( void )   = 0;
00056         virtual uint8_t GetPin ( void )         = 0;
00057         virtual uint8_t SetPinMode ( void )   = 0;
00058         virtual uint8_t SetPinResistor ( void )   = 0;
00059         virtual         ~InOut () = default;
00060 };
00061 #endif /* IN_OUT_H_ */
```

7.52 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/03-PinInterrupt/Pininterrupt.h File Reference

Clase para entradas con interrupciones por flanco.

```
#include <LPC845.h>
#include <Perifericos/01-Pin/Pin.h>
#include <Perifericos/02-Gpio/gpio.h>
```

Include dependency graph for Pininterrupt.h: This graph shows which files directly or indirectly include this file:

Classes

- class PinInterrupt

  Clase del objeto Pin_interrupt El objeto Pin_interrupt debe ser heredado por cualquier objeto que desee tener interupciones por pin.

Macros

- #define MAX_PININTERRUPT 8

Variables

- PinInterrupt * g_gpiohandler [MAX_PININTERRUPT]

7.52.1 Detailed Description

Clase para entradas con interrupciones por flanco.

Date

17 sep. 2022

Author

Grupo 4

7.52.2 Macro Definition Documentation

7.52.2.1 MAX_PININTERRUPT

```
#define MAX_PININTERRUPT 8
```

Cantidad máxima de interrupciones de pin

7.53 Pininterrupt.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009 #ifndef PININTERRUPT_H_
00010 #define PININTERRUPT_H_
00014
      /*****************************************************************************************************************************
00015  *** INCLUDES GLOBALES
00016
      *****************************************************************************************************************************
00017 #include <LPC845.h>
00018 #include <Perifericos/01-Pin/Pin.h>
00019 #include <Perifericos/02-Gpio/gpio.h>
00020
00021
      /*****************************************************************************************************************************
00022  *** DEFINES GLOBALES
00023
      *****************************************************************************************************************************
00024 #if defined (__cplusplus)
00025     extern "C" {
00026     void PININT0_IRQHandler(void);
00027     }
00028     extern "C" {
00029     void PININT1_IRQHandler(void);
00030     }
00031     extern "C" {
00032     void PININT2_IRQHandler(void);
00033     }
00034     extern "C" {
00035     void PININT3_IRQHandler(void);
00036     }
00037     extern "C" {
00038     void PININT4_IRQHandler(void);
00039     }
00040     extern "C" {
00041     void PININT5_IRQHandler(void);
00042     }
00043     extern "C" {
00044     void PININT6_IRQHandler(void);
00045     }
00046     extern "C" {
00047     void PININT7_IRQHandler(void);
00048     }
00049 #endif
00050
      /*****************************************************************************************************************************
00051  *** MACROS GLOBALES
00052
      *****************************************************************************************************************************
00053
00054
      /*****************************************************************************************************************************
00055  *** TIPO DE DATOS GLOBALES
00056
      *****************************************************************************************************************************
00057
00058
      /*****************************************************************************************************************************
00059  *** IMPLANTACION DE LA CLASE
00060
      *****************************************************************************************************************************
```

```
00066 class PinInterrupt : protected gpio
00067 {
00068 public:
00069     static  uint8_t m_cant;
00070     const   uint8_t m_interrupt_number;
00071     const   uint8_t m_interrput_mode;
00072
00073 public:
00075     #define MAX_PININTERRUPT    8
00076
00077 public:
00078                     PinInterrupt( port_t port , uint8_t bit , mode_t gpio_mode , activity_t activity ,
      uint8_t intrp_mode);
00079     void            EnableInterupt ( void );
00080     void            DisableInterupt ( void );
00081     void            PinInterrupt_Inicializar( void );
00082     virtual void    GpioHandler(void) = 0;
00083     virtual         ~PinInterrupt();
00084 private:
00085     void            PinInterrupt_Enable_clock( void );
00086
00087 };
00088
00090 extern PinInterrupt * g_gpiohandler[MAX_PININTERRUPT];
00091
00092 #endif /* PININTERRUPT_H_ */
```

7.54 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/04-Input/Input.cpp File Reference

funciones miembro de la clase Input
`#include <Drivers/Perifericos/04-Input/Input.h>`
Include dependency graph for Input.cpp:

Functions

- bool operator== (uint32_t val, Input &I)

  Sobrecarga de del operador de asignacion.

7.54.1 Detailed Description

funciones miembro de la clase Input

Date

27 may. 2022

Author

Ing. Marcelo Trujillo

7.54.2 Function Documentation

7.54.2.1 operator==()

```
bool operator== (
            uint32_t val,
            Input & I)
```

Sobrecarga de del operador de asignacion.

Parameters

in	*val*	Valor a comparar con el buffer.
in	*I*	Entrada a comparar.

Returns

true si la entrada esta en val.

7.55 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/04-Input/Input.h File Reference

funciones miembro de la clase Input
`#include <Perifericos/02-Gpio/gpio.h>`
`#include <Perifericos/06-Callback/Callback.h>`
Include dependency graph for Input.h: This graph shows which files directly or indirectly include this file:

Classes

- class Input

  Clase del objeto Input.

Macros

- #define MAX_BOUNCE 4

7.55.1 Detailed Description

funciones miembro de la clase Input

Date

27 may. 2022

Author

Ing. Marcelo Trujillo

7.55.2 Macro Definition Documentation

7.55.2.1 MAX_BOUNCE

`#define MAX_BOUNCE 4`
Cantidad de rebotes por defecto

7.56 Input.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef INPUTS_H_
00014 #define INPUTS_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include <Perifericos/02-Gpio/gpio.h>
00022 #include <Perifericos/06-Callback/Callback.h>
00023
00024
      /*****************************************************************************************************************************
00025  *** DEFINES GLOBALES
00026
      *****************************************************************************************************************************
```

```
00027
00028
      /**************************************************************************************************************************
00029  *** MACROS GLOBALES
00030
      **************************************************************************************************************************
00031
00032
      /**************************************************************************************************************************
00033  *** TIPO DE DATOS GLOBALES
00034
      **************************************************************************************************************************
00035
00036
      /**************************************************************************************************************************
00037  *** VARIABLES GLOBALES
00038
      **************************************************************************************************************************
00039
00040
      /**************************************************************************************************************************
00041  *** IMPLANTACION DE LA CLASE
00042
      **************************************************************************************************************************
      Input : protected gpio , public Callback
00049 {
00050     private:
00051         uint8_t m_BufferEntrada ;
00052         uint8_t m_MaxBounce;
00053         uint8_t m_CountBounce ;
00054     private:
00055         #define MAX_BOUNCE      4
00056
00057     public:
00058         Input( port_t puerto , uint8_t bit , mode_t modo , activity_t actividad = high , uint8_t
      MaxBounce = MAX_BOUNCE );
00059         void Inicializar ( void );
00060         uint8_t get ( void );
00061         void SWhandler ( void ) override;
00062         bool operator== ( uint8_t val );
00063         bool operator!= ( uint8_t val );
00071         friend bool operator==( uint32_t val , Input &I );
00072
00073         virtual ~Input();
00074     private:
00075         void SetBuffer ( void );
00076 };
00077
00078 #endif /* INPUTS_H_ */
```

7.57 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/05-Output/Output.cpp File Reference

Clase del tipo Output o salida digital.
`#include <Drivers/Perifericos/05-Output/Output.h>`
Include dependency graph for Output.cpp:

7.57.1 Detailed Description

Clase del tipo Output o salida digital.

Date

12 jul. 2022

Author

Ing. Marcelo Trujillo

7.58 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/05-Output/Output.h File Reference

Clase del tipo Output o salida digital.

```
#include <Perifericos/02-Gpio/gpio.h>
#include <Perifericos/06-Callback/Callback.h>
```

Include dependency graph for Output.h: This graph shows which files directly or indirectly include this file:

Classes

- class Output

  Clase del objeto outputs.

7.58.1 Detailed Description

Clase del tipo Output o salida digital.

Date

12 jul. 2022

Author

Ing. Marcelo Trujillo

7.59 Output.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef OUTPUTS_H_
00014 #define OUTPUTS_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include <Perifericos/02-Gpio/gpio.h>
00022 #include <Perifericos/06-Callback/Callback.h>
00023
00024
      /*****************************************************************************************************************************
00025  *** DEFINES GLOBALES
00026
      *****************************************************************************************************************************
00027
00028
      /*****************************************************************************************************************************
00029  *** MACROS GLOBALES
00030
      *****************************************************************************************************************************
00031
00032
      /*****************************************************************************************************************************
00033  *** TIPO DE DATOS GLOBALES
00034
      *****************************************************************************************************************************
00035
00036
      /*****************************************************************************************************************************
00037  *** VARIABLES GLOBALES
00038
      *****************************************************************************************************************************
00039
00040
      /*****************************************************************************************************************************
00041  *** IMPLANTACION DE UNA CLASE
00042
      *****************************************************************************************************************************
00049 class Output : protected gpio , public Callback
00050 {
00051     uint8_t m_buffer;
00052
00053     public:
```

```
00054         Output( port_t puerto , uint8_t bit , mode_t modo , activity_t actividad = high , uint8_t
      estado = on );
00055         int8_t On ( void );
00056         int8_t Off ( void );
00057         int8_t SetUp( void );
00058 
00059         Output& operator= ( uint8_t estado );
00060         bool operator== ( uint8_t a );
00061         void SWhandler ( void ) override;
00062 
00063         virtual ~Output();
00064 };
00065 
00066 #endif /* OUTPUTS_H_ */
```

7.60 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/06-Callback/Callback.cpp File Reference

funciones miembro de la clase Callback
`#include <Perifericos/06-Callback/Callback.h>`
Include dependency graph for Callback.cpp:

Variables

- vector< Callback * > g_Handler

7.60.1 Detailed Description

funciones miembro de la clase Callback

Date

04 ene. 2025

Author

Tecnico Martinez Agustin

7.61 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/06-Callback/Callback.h File Reference

Clase virtual pura. Esta clase debe ser heredada por las clases que se tienen que enganchar del Systick Timer.
`#include <LPC845.h>`
`#include <vector>`
`#include <systick.h>`
Include dependency graph for Callback.h: This graph shows which files directly or indirectly include this file:

Classes

- class Callback

  Clase del objeto Callback.

Macros

- #define TICK_SECONDS(x)
- #define TICK_MILISECONDS(x)
- #define TICK_MICROSECONDS(x)

Variables

- vector< Callback * > g_Handler

7.61.1 Detailed Description

Clase virtual pura. Esta clase debe ser heredada por las clases que se tienen que enganchar del Systick Timer.

Date

4 may. 2022

Author

Ing. Marcelo Trujillo

7.61.2 Macro Definition Documentation

7.61.2.1 TICK_MICROSECONDS

```
#define TICK_MICROSECONDS(
              x)
```

Value:

```
((x) * (g_systick_freq/1000000))
```

MACRO expresion to obtain the corresponding TICKS for "x" microseconds

7.61.2.2 TICK_MILISECONDS

```
#define TICK_MILISECONDS(
              x)
```

Value:

```
((x) * (g_systick_freq/1000))
```

MACRO expresion to obtain the corresponding TICKS for "x" miliseconds

7.61.2.3 TICK_SECONDS

```
#define TICK_SECONDS(
              x)
```

Value:

```
((x) * g_systick_freq)
```

MACRO expresion to obtain the corresponding TICKS for "x" Seconds

7.62 Callback.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009 #ifndef CALLBACK_H_
00010 #define CALLBACK_H_
00014
      /*****************************************************************************************************************************
00015  *** INCLUDES GLOBALES
00016
      *****************************************************************************************************************************
00017 #include <LPC845.h>
00018 #include <vector>
00019 #include <systick.h>
00020 using namespace std;
00021
00022
      /*****************************************************************************************************************************
00023  *** DEFINES GLOBALES
00024
      *****************************************************************************************************************************
00025
00026
      /*****************************************************************************************************************************
00027  *** MACROS GLOBALES
00028
      *****************************************************************************************************************************
00029
00030
      /*****************************************************************************************************************************
```

```
00031  *** IMPLANTACION DE LA CLASE
00032
       ***********************************************************************************************************************
00039 class Callback
00040 {
00041     protected:
00043         #define TICK_SECONDS(x)          ((x) * g_systick_freq)
00045         #define TICK_MILISECONDS(x)      ((x) * (g_systick_freq/1000))
00047         #define TICK_MICROSECONDS(x)     ((x) * (g_systick_freq/1000000))
00048     public:
00049         Callback() = default;
00050         void SetInterrupt();
00051         void UnSetInterrupt();
00052
00056         virtual void SWhandler( void ) = 0;
00057
00058         virtual ~Callback( ) = default;
00059 };
00060
00062 extern vector <Callback*> g_Handler;
00063
00064 #endif /* CALLBACK_H_ */
```

7.63 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/07-Timer/Timer.h File Reference

Clase para creacion de temporizadores.

```
#include <Perifericos/06-Callback/Callback.h>
#include "tipos.h"
```

Include dependency graph for Timer.h: This graph shows which files directly or indirectly include this file:

Classes

- class Timer

  Clase del objeto timer.

Typedefs

- typedef void(∗ Timer_Handler) (void)

7.63.1 Detailed Description

Clase para creacion de temporizadores.

Date

4 may. 2022

Author

Ing. Marcelo Trujillo

7.64 Timer.h

Go to the documentation of this file.

```
00001
       ***********************************************************************************************************************
00009 #ifndef TIMER_H
00010 #define TIMER_H
00014
       ***********************************************************************************************************************
00015  *** INCLUDES GLOBALES
00016
       ***********************************************************************************************************************
00017 #include <Perifericos/06-Callback/Callback.h>
00018 #include "tipos.h"
00019
00020
       ***********************************************************************************************************************
```

```
00021  *** DEFINES GLOBALES
00022
       ********************************************************************************************************************************
00023
00024
       /*******************************************************************************************************************************
00025  *** MACROS GLOBALES
00026
       ********************************************************************************************************************************
00027
00028
       /*******************************************************************************************************************************
00029  *** TIPO DE DATOS GLOBALES
00030
       ********************************************************************************************************************************
00031 typedef void (*Timer_Handler)(void);
00032
00033
       /*******************************************************************************************************************************
00034  *** IMPLANTACION DE LA CLASE
00035
       ********************************************************************************************************************************
00042 class Timer : public Callback
00043 {
00044     private:
00045         enum        ticks_t             { DECIMAS = 100 , SEGUNDOS = 10 , MINUTOS  = 60 };
00046     public:
00048         typedef enum        bases_t   { DEC , SEG , MIN , HOR } bases_t;
00049         enum        erroresTimers_t   { errorTimer , OKtimers };
00050         enum        standby_t           { RUN , PAUSE };
00051
00052     protected:
00053         volatile uint32_t   m_TmrRun;
00054         volatile bool       m_TmrEvent;
00055         void        (* m_TmrHandler ) (void);
00056         volatile bool       m_TmrStandBy ;
00057         volatile bases_t     m_TmrBase ;
00058
00059     public:
00060         Timer( ) ;
00061         Timer ( const bases_t base , const Timer_Handler handler = nullptr );
00062         void        TimerStart ( uint32_t time, const Timer_Handler handler , const bases_t base );
00063         void        SetTimer(  uint32_t time );
00064         uint32_t    GetTimer( void ) const;
00065         void        StandByTimer( const uint8_t accion );
00066         void        SetTimerBase( const bases_t base );
00067         void        TimerStop( void );
00068         uint32_t    GetTmrRun( void );
00069         void        SetTmrEvent( void );
00070         void        ClrTmrEvent( void );
00071         bool        GetTmrEvent( void );
00072         bool        GetmrStandBy( void );
00073         void        SetmrStandBy( uint8_t accion);
00074         void        SetTmrHandler( void );
00075         void        TimerStart( uint32_t time );
00076
00077         Timer&         operator=( uint32_t t );
00078         bool        operator!( );
00079         explicit    operator bool () ;
00080         bool        operator==( uint32_t t );
00081
00082         // por el hecho de haber convertido el operador bool como explicit
00083         // me obligo a realizar las funciones amigas del operador==
00084         // con sus dos prototipos, porque dejo de aceptar
00085         // la promocion automatica de tipos
00092         friend bool     operator==( uint32_t t , Timer &T );
00093         // Implementacion de funcion virtual pura heredada
00094         void            SWhandler( void );
00095         int8_t TmrEvent  ( void );
00096
00097         virtual         ~Timer();
00098 };
00099
00100 #endif /* TIMER_H */
```

7.65 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/08-Timers/Timers.cpp File Reference

funciones miembro de la clase timers

`#include <Perifericos/08-Timers/Timers.h>`

Include dependency graph for Timers.cpp:

7.65.1 Detailed Description

funciones miembro de la clase timers
Descripcion del modulo.

Date

27 may. 2022

Author

Ing. Marcelo Trujillo

Date

10 jul. 2022

Author

Ing. Marcelo Trujillo

7.66 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/08-Timers/Timers.h File Reference

Breve descripción del objetivo del Módulo.
`#include <Perifericos/07-Timer/Timer.h>`
Include dependency graph for Timers.h: This graph shows which files directly or indirectly include this file:

Classes

- class timers

  Clase del objeto timers El objeto timers permite agrupar todos los timers y ejecutarlos de una sola pasada. Permite ahorrar código.

7.66.1 Detailed Description

Breve descripción del objetivo del Módulo.

Date

10 jul. 2022

Author

Ing. Marcelo Trujillo

7.67 Timers.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef TIMERS_H_
00014 #define TIMERS_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
```

```
00021 #include <Perifericos/07-Timer/Timer.h>
00022
00023
      /*****************************************************************************************************************************
00024  *** DEFINES GLOBALES
00025
      *****************************************************************************************************************************
00026
00027
      /*****************************************************************************************************************************
00028  *** MACROS GLOBALES
00029
      *****************************************************************************************************************************
00030
00031
      /*****************************************************************************************************************************
00032  *** TIPO DE DATOS GLOBALES
00033
      *****************************************************************************************************************************
00034
00035
      /*****************************************************************************************************************************
00036  *** VARIABLES GLOBALES
00037
      *****************************************************************************************************************************
00038
00039
      /*****************************************************************************************************************************
00040  *** IMPLANTACION DE UNA CLASE
00041
      *****************************************************************************************************************************
00047 class timers
00048 {
00049     private:
00050         vector <Timer* > m_timers;
00051     public:
00052                     timers();
00053         timers &  operator«( Timer *t );
00054         void        TmrEvent( void );
00055         virtual     ~timers() = default;
00056 };
00057
00058 #endif /* TIMERS_H_ */
```

7.68 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/09-MultiRateTimer/MRThandler.cpp File Reference

Handler del timer MRT.
`#include <Drivers/Perifericos/09-MultiRateTimer/MRThandler.h>`
Include dependency graph for MRThandler.cpp:

Variables

- std::vector< MRThandler * > g_MRThandler

7.68.1 Detailed Description

Handler del timer MRT.

Date

2 sep. 2022

Author

Técnico Martinez Agustin

7.69 MRThandler.h

```
00001
      /*****************************************************************************************************************************
00009
```

```
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef MRTHANDLER_H_
00014 #define MRTHANDLER_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES
00020
      *****************************************************************************************************************************
00021 #include "tipos.h"
00022 #include "LPC845.h"
00023 #include "vector"
00024
      /*****************************************************************************************************************************
00025  *** DEFINES GLOBALES
00026
      *****************************************************************************************************************************
00027 #if defined (__cplusplus)
00028     extern "C" {
00029     void MRT_IRQHandler(void);
00030     }
00031 #endif
00032
      /*****************************************************************************************************************************
00033  *** MACROS GLOBALES
00034
      *****************************************************************************************************************************
00035 namespace type_MRT
00036 {
00037     typedef enum { CHANNEL_0 = 0, CHANNEL_1, CHANNEL_2, CHANNEL_3 } MRT_timer_channels;
00038     typedef enum { REPEAT = 0, ONE_SHOT, ONE_SHOT_BUS, COUNTER } MRT_MODES ;
00040     #define MAX_MRT_CHANNEL     4
00041 }
00042 using namespace type_MRT;
00043
      /*****************************************************************************************************************************
00044  *** TIPO DE DATOS GLOBALES
00045
      *****************************************************************************************************************************
00046
00047
      /*****************************************************************************************************************************
00048  *** VARIABLES GLOBALES
00049
      *****************************************************************************************************************************
00050
00051
      /*****************************************************************************************************************************
00052  *** IMPLANTACION DE LA CLASE
00053
      *****************************************************************************************************************************
00059 class MRThandler
00060 {
00061     private:
00062         static bool m_first[MAX_MRT_CHANNEL];
00063     public:
00064         const MRT_timer_channels m_timer_channel;
00065     public:
00066                 MRThandler( MRT_timer_channels _timer_number , MRT_MODES mode );
00067         virtual ~MRThandler() = default;
00068
00072         virtual void    Handler ( void ) = 0;
00073         void            EneableInterrupt ( void );
00074         void            DisableInterrupt ( void );
00075
00076     protected:
00077         void        MRT_reset_time ( void );
00078         uint32_t    MRT_get_time ( void );
00079     private:
00080         void    MRT_Inicializar ( MRT_timer_channels timer , MRT_MODES mode  );
00081         void    MRT_Reset ( void );
00082 };
00083
00085 extern std::vector <MRThandler *> g_MRThandler;
00086
00087 #endif /* MRTHANDLER_H_ */
```

7.70 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/10-SCTimer/SCtimer.cpp File Reference

Salida autónoma temporizada.
`#include <Drivers/Perifericos/10-SCTimer/SCtimer.h>`
Include dependency graph for SCtimer.cpp:

7.70.1 Detailed Description

Salida autónoma temporizada.

Date

7 oct. 2022

Author

Grupo 4

7.71 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/10-SCTimer/SCtimer.h File Reference

Salida autónoma temporizada.
`#include "LPC845.h"`
Include dependency graph for SCtimer.h: This graph shows which files directly or indirectly include this file:

Classes

- class SCtimer

  Clase del objeto SCtimer El objeto SCtimer debe ser heredado por quienes desen utilizar las interrucpiones o funcionalidades del SCtimer.

7.71.1 Detailed Description

Salida autónoma temporizada.

Date

7 oct. 2022

Author

Grupo 4

7.72 SCtimer.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef SCTIMER_H_
00014 #define SCTIMER_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include "LPC845.h"
```

```
00022
      /*****************************************************************************************************************************
00023  *** DEFINES GLOBALES
00024
      *****************************************************************************************************************************
00025
00026
      /*****************************************************************************************************************************
00027  *** MACROS GLOBALES
00028
      *****************************************************************************************************************************
00029
00030
      /*****************************************************************************************************************************
00031  *** TIPO DE DATOS GLOBALES
00032
      *****************************************************************************************************************************
00033
00034
      /*****************************************************************************************************************************
00035  *** IMPLANTACION DE LA CLASE
00036
      *****************************************************************************************************************************
00042 class SCtimer {
00043 public:
00044             SCtimer ();
00045     void    SetTime ( uint32_t time , uint32_t channel );
00046     void    SetUnify ( bool a );
00047     void    SetAutoLimit ( bool a );
00048     void    SetSwitchMatrizSCTOUT ( uint8_t bit , uint8_t port , uint8_t out_number);
00049     void    StartTimer ( void );
00050     void    StopTimer ( void );
00051
00052     virtual ~SCtimer();
00053 };
00054
00055 #endif /* SCTIMER_H_ */
```

7.73 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/11-DAC/DAC.h File Reference

Objeto DAC.

```
#include <Perifericos/01-Pin/Pin.h>
#include "LPC845.h"
#include "tipos.h"
```

Include dependency graph for DAC.h: This graph shows which files directly or indirectly include this file:

Classes

- class DAC

  Clase del objeto DAC FUNCIONAMIENTO: Realiza una conversion digital->analógica en un rango desde 0 hasta max_range. El rango real del LPC845 va de 0 hasta 1023. Se realiza una conversion lineal entre el rango del dispositivo y el utilizado por el usuario. NO USAR EL CHANNEL 1. El canal existe segun datasheet pero los registros son vagos y poco explicativos. Corresponde al PINENABLE. Recomendado utilizar solo el CHANNEL 0.

Macros

- #define MAX_DAC_CHANNEL (2)
- #define MAX_DAC_VALUE (0x3FF)

7.73.1 Detailed Description

Objeto DAC.

Date

12 ene. 2023

Author

Técnico Martinez Agustin

7.73.2 Macro Definition Documentation

7.73.2.1 MAX_DAC_CHANNEL

```
#define MAX_DAC_CHANNEL (2)
```

Máximos canales del DAC

7.73.2.2 MAX_DAC_VALUE

```
#define MAX_DAC_VALUE (0x3FF)
```

Máximo valor del DAC por defecto

7.74 DAC.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef DAC_H_
00014 #define DAC_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include <Perifericos/01-Pin/Pin.h>
00022 #include "LPC845.h"
00023 #include "tipos.h"
00024
      /*****************************************************************************************************************************
00025  *** DEFINES GLOBALES
00026
      *****************************************************************************************************************************
00027
00028
      /*****************************************************************************************************************************
00029  *** MACROS GLOBALES
00030
      *****************************************************************************************************************************
00031
00032
      /*****************************************************************************************************************************
00033  *** TIPO DE DATOS GLOBALES
00034
      *****************************************************************************************************************************
00035
00036
      /*****************************************************************************************************************************
00037  *** VARIABLES GLOBALES
00038
      *****************************************************************************************************************************
00039
00040
      /*****************************************************************************************************************************
00041  *** IMPLANTACION DE LA CLASE
00042
      *****************************************************************************************************************************
00052 class DAC : protected Pin
00053 {
00054     public:
00056         #define MAX_DAC_CHANNEL     (2)
00058         #define MAX_DAC_VALUE           (0x3FF)
00060         typedef enum dac_channel { DAC_CHANNEL_0 = 0 , DAC_CHANNEL_1 = 1 } dac_channel;
00062         typedef enum dac_error { ERROR = -1 , OK = 0 } dac_error;
00063
00064     private:
00066         enum OFFSET { SWM_DACOUT0 = 26 , SWM_DACOUT1 = 27 , SYS_DACOUT0 = 27 , SYS_DACOUT1 = 1 ,
      IOCON_DAC = 16};
00067
00068     private:
```

```
00069         const dac_channel m_dac_channel;
00070         uint32_t m_buffer;
00071         uint32_t m_max_range;
00072         dac_error m_error;
00073 
00074     public:
00075         DAC( dac_channel channel , uint32_t max_range = MAX_DAC_VALUE );
00076         DAC::dac_error Inicializar ( void );
00077         void Set ( uint32_t val );
00078         uint32_t Get ( void ) const;
00079 
00080         void SetMaxRange ( uint32_t max_range );
00081         uint32_t GetMaxRange ( void ) const;
00082 
00083         DAC& operator= ( uint32_t val );
00084         bool operator== ( uint32_t val ) const;
00085         bool operator< ( uint32_t val ) const;
00086         bool operator<= ( uint32_t val ) const;
00087         bool operator> ( uint32_t val ) const;
00088         bool operator>= ( uint32_t val ) const;
00089         bool operator!= ( uint32_t val ) const;
00090 
00091         virtual ~DAC();
00092 
00093     private:
00094         void PowerDAC( void );
00095         void EnableClock ( void );
00096         void EnableSWM ( void );
00097         void EnableIOCONDAC( void );
00098 
00099         void UnPowerDAC ( void );
00100         void DisableClock ( void );
00101         void DisableSWM ( void );
00102         void DisableIOCONDAC( void );
00103 };
00104 
00105 #endif /* DAC_H_ */
```

7.75 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/12-ADC/ADC.cpp File Reference

Breve descripcion del modulo.

`#include <Drivers/Perifericos/12-ADC/ADC.h>`

Include dependency graph for ADC.cpp:

7.76 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/12-ADC/ADC.h File Reference

Breve descripción del Módulo.

`#include <Drivers/Perifericos/12-ADC/ADCGroup.h>`

`#include <Perifericos/01-Pin/Pin.h>`

Include dependency graph for ADC.h: This graph shows which files directly or indirectly include this file:

Classes

- class ADC

  Clase del objeto ADC FUNCIONAMIENTO: La clase ADC utiliza el ADCGroup para poder ser manejada de forma individual por cada pata. Se pueden crear tantos objetos como canales del ADC existen. La configuración de conversión se realiza automáticamente con el primer objeto ADC creado, el resto no necesita recibir ninguna frecuencia de clock o muestreo.

7.76.1 Detailed Description

Breve descripción del Módulo.

Date

22 feb. 2023

Author

Técnico Martinez Agustín

Version

v1.0

7.77 ADC.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00010
00011
      /*****************************************************************************************************************************
00012  *** MODULO
00013
      *****************************************************************************************************************************
00014 #ifndef ADC_H_
00015 #define ADC_H_
00019
      /*****************************************************************************************************************************
00020  *** INCLUDES GLOBALES
00021
      *****************************************************************************************************************************
00022 #include <Drivers/Perifericos/12-ADC/ADCGroup.h>
00023 #include <Perifericos/01-Pin/Pin.h>
00024
      /*****************************************************************************************************************************
00025  *** DEFINES GLOBALES
00026
      *****************************************************************************************************************************
00027
00028
      /*****************************************************************************************************************************
00029  *** MACROS GLOBALES
00030
      *****************************************************************************************************************************
00031
00032
      /*****************************************************************************************************************************
00033  *** TIPO DE DATOS GLOBALES
00034
      *****************************************************************************************************************************
00035
00036
      /*****************************************************************************************************************************
00037  *** VARIABLES GLOBALES
00038
      *****************************************************************************************************************************
00039
00040
      /*****************************************************************************************************************************
00041  *** IMPLANTACION DE LA CLASE
00042
      *****************************************************************************************************************************
00051 class ADC : public Pin
00052 {
00053 private:
00054     static ADC_Group* m_global_adc;
00055     uint8_t m_channel;
00056 public:
00057     ADC( uint8_t _channel , uint32_t _clk_freq = 0 , uint32_t _sample_rate = 0 );
00058     int32_t Get ( void );
00059     bool IsResultReady ( void );
00060     void Trigger ( void );
00061     void Inicializar ( void );
00062     virtual ~ADC() = default;
00063 };
00064
00065 #endif /* ADC_H_ */
```

7.78 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/12-ADC/ADCGroup.cpp File Reference

Objeto DAC.

`#include <Drivers/Perifericos/12-ADC/ADCGroup.h>`
Include dependency graph for ADCGroup.cpp:

Variables

- ADC_Group * **g_adc** = nullptr
  vector de interrupciones ADC

7.78.1 Detailed Description

Objeto DAC.

Date

13 nov. 2022

Author

Federico

7.79 ADCGroup.h

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef ADCGROUP_H_
00014 #define ADCGROUP_H_
00018
      /*****************************************************************************************************************************
00019  *** INCLUDES GLOBALES
00020
      *****************************************************************************************************************************
00021 #include "tipos.h"
00022 #include "LPC845.h"
00023
      /*****************************************************************************************************************************
00024  *** DEFINES GLOBALES
00025
      *****************************************************************************************************************************
00026 #if defined (__cplusplus)
00027     extern "C" {
00028     void ADC_SEQA_IRQHandler ( void );
00029     }
00030     extern "C" {
00031     void ADC_SEQB_IRQHandler ( void );
00032     }
00033     extern "C" {
00034     void ADC_THCMP_IRQHandler ( void );
00035     }
00036     extern "C" {
00037     void ADC_OVR_IRQHandler ( void );
00038     }
00039     extern "C" {
00040     void UART4_IRQHandler ( void );
00041     }
00042 #endif
00043
00044
      /*****************************************************************************************************************************
00045  *** MACROS GLOBALES
00046
      *****************************************************************************************************************************
00047
00048
      /*****************************************************************************************************************************
00049  *** TIPO DE DATOS GLOBALES
00050
      *****************************************************************************************************************************
00051
00052
      /*****************************************************************************************************************************
00053  *** VARIABLES GLOBALES
```

```
00054
    ********************************************************************************************************************************************
00055
00056
    /*****************************************************************************************************************************************
00057  *** IMPLANTACION DE LA CLASE
00058
    ********************************************************************************************************************************************
00066 class ADC_Group
00067 {
00068     public:
00070     #define MAX_ADC_CHANNELS    (12)
00071
00073     typedef enum adc_isr { SEQA_ISR , SEQB_ISR , THCMP_ISR , OVR_ISR } adc_isr;
00075     typedef enum irq_source_inten{ ADC_SEQA_IRQ_INTEN = 0, ADC_SEQB_IRQ_INTEN = 1, ADC_OVR_IRQ_INTEN =
    2, INVALID_IRQ_INTEN = 0xFF} irq_source_inten;
00076     //Todas las de threshold no serán tenidas en cuenta
00078     typedef enum error_t { OK = 0 , ERROR = -1 } error_t;
00079
00080     private:
00082     #define CLOCKS_PER_SAMPLE   (25)
00084     #define CLK_500KHz          (500000)
00086     typedef enum irq_source_nvic{ADC_SEQA_IRQ = 16, ADC_SEQB_IRQ = 17, ADC_THCMP_IRQ = 18, ADC_OVR_IRQ
    = 19, INVALID_IRQ = 0xFF } irq_source_nvic;
00088     typedef enum trm_voltage_config { HIGH_VOLTAGE = 0 , LOW_VOLTAGE = 1 } trm_voltage_config;
00090     typedef enum conversion_mode { CONVERSION_INTERRUPT = 0 , SEQUENCE_INTERRUPT = 1} conversion_mode;
00091
00092     private:
00093         uint32_t    m_sample_rate;
00094         uint32_t    m_clk_freq;
00095         uint16_t    m_enabled_channels;
00096         uint32_t    m_result[MAX_ADC_CHANNELS];
00097         bool        m_result_ready[MAX_ADC_CHANNELS];
00098
00099     public:
00100             ADC_Group( uint32_t clk_freq , uint32_t sample_rate , bool init_channel0 = false );
00101         virtual ~ADC_Group() = default;
00102         void    Inicializar( void );
00103
00104         void    SetLowPowerMode  ( bool low_power );
00105         void    SetSampleRate  ( void );
00106
00107         void    EnableIrq  ( irq_source_inten irq );
00108         void    DisableIrq    ( irq_source_inten irq );
00109
00110         ADC_Group::error_t    InitADCChanel  ( uint8_t channel );    //Adds a channel to the ADC
00111         ADC_Group::error_t    RemoveADCChanel  ( uint8_t channel );    //Removes a channel to the ADC
00112         void                TriggerStartSeqA( void );
00113
00114         int32_t GetValue        ( uint8_t channel );
00115         bool    IsResultReady  ( uint8_t channel ) const;
00116         void    Handler          ( adc_isr isr );
00117
00118     private:
00119         void            InitADC ();
00120         uint32_t        CalculateDivisor ( uint32_t sample_rate );
00121         void            SetADCVoltage ( trm_voltage_config config );
00122         void            EnableADCPower ( void );
00123         void            EnableADCClock ( void );
00124         void            ConfigSWM ( uint8_t channel , bool enable );
00125         void            CalibrateADC ();
00126         void            ADCConfig ();
00127         void            AddChannel_to_SequenceA ( uint8_t channel );
00128         void            AddChannel_to_SequenceB ( uint8_t channel );
00129         void            RemoveChannelOfSequenceA ( uint8_t channel );
00130         void            RemoveChannelOfSequenceB ( uint8_t channel );
00131         void            SetSeqAMode ( conversion_mode mode );
00132         void            SetSeqBMode ( conversion_mode mode );
00133         void            EnableSeqA ( void );
00134         void            EnableSeqB ( void );
00135         void            DisableSeqA ( void );
00136         void            DisableSeqB ( void );
00137         void            SetUpSeqA ( void );
00138         void            EnableNvicADCInterrupt ( irq_source_nvic source );
00139         void            DisableNvicADCInterrupt ( irq_source_nvic source );
00140         irq_source_nvic GetNvicIrq ( irq_source_inten irq );
00141         void            handlerSeqA ( void );
00142         uint16_t        GetResult ( uint8_t channel );
00143 };
00144
00145 #endif /* ADCGROUP_H */
```

7.80 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/13-UART/ComunicacionAsincronica.h File Reference

Objeto base para la creacion de comunicaciones asincrónicas.
`#include "tipos.h"`
Include dependency graph for ComunicacionAsincronica.h: This graph shows which files directly or indirectly include this file:

Classes

- class ComunicacionAsincronica

  Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de UART.

7.80.1 Detailed Description

Objeto base para la creacion de comunicaciones asincrónicas.

Date

5 oct. 2022

Author

Ing. Marcelo Trujillo

7.81 ComunicacionAsincronica.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013
00014 #ifndef COMUNICACIONASINCRONICA_H_
00015 #define COMUNICACIONASINCRONICA_H_
00019
      /*****************************************************************************************************************************
00020  *** INCLUDES GLOBALES
00021
      *****************************************************************************************************************************
00022 #include "tipos.h"
00023
00024
      /*****************************************************************************************************************************
00025  *** DEFINES GLOBALES
00026
      *****************************************************************************************************************************
00027
00028
      /*****************************************************************************************************************************
00029  *** MACROS GLOBALES
00030
      *****************************************************************************************************************************
00031
00032
      /*****************************************************************************************************************************
00033  *** TIPO DE DATOS GLOBALES
00034
      *****************************************************************************************************************************
00035
00036
      /*****************************************************************************************************************************
00037  *** VARIABLES GLOBALES
00038
      *****************************************************************************************************************************
00039
00040
      /*****************************************************************************************************************************
00041  *** IMPLANTACION DE UNA CLASE
```

```
00042
     ***********************************************************************************************************************************************
00048 class ComunicacionAsincronica
00049 {
00050     public:
00051                         ComunicacionAsincronica() = default;
00052         virtual void    Transmit ( const char * msg) = 0;
00053         virtual void    Transmit ( const void * msg , uint32_t n ) = 0;
00054         virtual void*   Message ( void * msg , uint32_t n ) = 0;
00055         virtual void    UART_IRQHandler (void) = 0;
00056         virtual         ~ComunicacionAsincronica() = default;
00057     protected:
00058         virtual void    pushRx ( uint8_t dato ) = 0 ;
00059         virtual uint8_t popRx (uint8_t * dato ) = 0 ;
00060         virtual void    pushTx ( uint8_t dato ) = 0 ;
00061         virtual uint8_t popTx (uint8_t * dato ) = 0 ;
00062 };
00063
00064 #endif /* COMUNICACIONASINCRONICA_H_ */
```

7.82 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/13-UART/Uart.cpp File Reference

Descripcion del modulo.
`#include <Drivers/Perifericos/13-UART/Uart.h>`
Include dependency graph for Uart.cpp:

Variables

- ComunicacionAsincronica * g_usart [5]

7.82.1 Detailed Description

Descripcion del modulo.

Date

5 oct. 2022

Author

Ing. Marcelo Trujillo

7.82.2 Variable Documentation

7.82.2.1 g_usart

`ComunicacionAsincronica* g_usart[5]`
Vector de UART s

7.83 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/13-UART/Uart.h File Reference

Breve descripción del objetivo del Módulo.
`#include <Drivers/Perifericos/13-UART/ComunicacionAsincronica.h>`
`#include "LPC845.h"`
`#include <Drivers/Perifericos/01-Pin/Pin.h>`
Include dependency graph for Uart.h: This graph shows which files directly or indirectly include this file:

Classes

- class Uart

  Clase del objeto uart El objeto uart genera una comunicación asincrónica de tipo UART.

7.83.1 Detailed Description

Breve descripción del objetivo del Módulo.

Date

5 oct. 2022

Author

Ing. Marcelo Trujillo

7.84 Uart.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************
00009
00010
      /*****************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************
00013 #ifndef UART_H_
00014 #define UART_H_
00018
00019
      /*****************************************************************************************************************
00020  *** INCLUDES GLOBALES
00021
      *****************************************************************************************************************
00022 #include <Drivers/Perifericos/13-UART/ComunicacionAsincronica.h>
00023 #include "LPC845.h"
00024 #include <Drivers/Perifericos/01-Pin/Pin.h>
00025
      /*****************************************************************************************************************
00026  *** DEFINES GLOBALES
00027
      *****************************************************************************************************************
00028 #if defined (__cplusplus)
00029     extern "C" {
00030     void UART0_IRQHandler ( void );
00031     }
00032     extern "C" {
00033     void UART1_IRQHandler ( void );
00034     }
00035     extern "C" {
00036     void UART2_IRQHandler ( void );
00037     }
00038     extern "C" {
00039     void UART3_IRQHandler ( void );
00040     }
00041     extern "C" {
00042     void UART4_IRQHandler ( void );
00043     }
00044 #endif
00045
      /*****************************************************************************************************************
00046  *** MACROS GLOBALES
00047
      *****************************************************************************************************************
00048
00049
      /*****************************************************************************************************************
00050  *** TIPO DE DATOS GLOBALES
00051
      *****************************************************************************************************************
00052
00053
      /*****************************************************************************************************************
00054  *** VARIABLES GLOBALES
00055
      *****************************************************************************************************************
00056
00057
      /*****************************************************************************************************************
00058  *** IMPLANTACION DE UNA CLASE
00059
      *****************************************************************************************************************
00065 class Uart : public ComunicacionAsincronica
00066 {
```

```
00067     public:
00069         typedef enum  { NoParidad , par = 2, impar} paridad_t;
00071         typedef enum  { siete_bits , ocho_bits } bits_de_datos;
00074         #define PORT_TX_USB     Pin::port0
00075         #define PIN_TX_USB      25
00076         #define PORT_RX_USB     Pin::port0
00077         #define PIN_RX_USB      24
00078         #define USART_USB       USART0
00080
00081     private:
00082         const Pin m_tx;
00083         const Pin m_rx;
00084         USART_Type* m_usart;
00085         uint8_t*    m_bufferRX;
00086         uint32_t    m_inxRxIn , m_inxRxOut , m_maxRx;
00087         uint8_t*    m_bufferTX;
00088         uint32_t    m_inxTxIn , m_inxTxOut , m_maxTx;
00089         bool        m_flagTx;
00090
00091     public:
00092         Uart( Pin::port_t portTx , uint8_t pinTx , Pin::port_t portRx , uint8_t pinRx ,
00093                 USART_Type * usart , uint32_t baudrate , bits_de_datos BitsDeDatos, paridad_t paridad
      ,
00094                 uint32_t maxRx , uint32_t maxTx);
00095         void Transmit ( const char * msg) override;
00096         void Transmit ( const void * msg , uint32_t n ) override;
00097         void* Message ( void * msg , uint32_t n ) override;
00098         void SetBaudRate ( uint32_t baudrate );
00099
00100         ~Uart() { };
00101     private:
00102         void EnableSW ( void );
00103         void EnableClock ( void );
00104         void Config ( uint32_t baudrate , bits_de_datos BitsDeDatos , paridad_t paridad );
00105         void UART_IRQHandler ( void ) override;
00106
00107         void pushRx ( uint8_t dato ) override;
00108         uint8_t popRx ( uint8_t * dato ) override;
00109         void pushTx ( uint8_t dato ) override;
00110         uint8_t popTx ( uint8_t * dato ) override;
00111         void EnableInterupt ( void );
00112         void DisableInterupt ( void );
00113 };
00114
00115 #endif /* UART_H_ */
```

7.85 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/14-ComSincronica/ComunicacionSincronica.h File Reference

Objeto base para la creacion de comunicaciones sincrónicas.

```
#include <Perifericos/01-Pin/Pin.h>
#include "tipos.h"
```

Include dependency graph for ComunicacionSincronica.h: This graph shows which files directly or indirectly include this file:

Classes

- class ComunicacionSincronica

  Clase del objeto ComunicacionAsincronica Clase abstracta pura para la generación de comunicaciones sincrónicas como la I2C o la SPI.

7.85.1 Detailed Description

Objeto base para la creacion de comunicaciones sincrónicas.

Date

5 oct. 2022

Author

Tecnico Martinez Agustin

7.86 ComunicacionSincronica.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013
00014 #ifndef COMUNICACIONSINCRONICA_H_
00015 #define COMUNICACIONSINCRONICA_H_
00019
      /*****************************************************************************************************************************
00020  *** INCLUDES GLOBALES
00021
      *****************************************************************************************************************************
00022 #include <Perifericos/01-Pin/Pin.h>
00023 #include "tipos.h"
00024
00025
      /*****************************************************************************************************************************
00026  *** DEFINES GLOBALES
00027
      *****************************************************************************************************************************
00028
00029
      /*****************************************************************************************************************************
00030  *** MACROS GLOBALES
00031
      *****************************************************************************************************************************
00032
00033
      /*****************************************************************************************************************************
00034  *** TIPO DE DATOS GLOBALES
00035
      *****************************************************************************************************************************
00036
00037
      /*****************************************************************************************************************************
00038  *** VARIABLES GLOBALES
00039
      *****************************************************************************************************************************
00040
00041
      /*****************************************************************************************************************************
00042  *** IMPLANTACION DE UNA CLASE
00043
      *****************************************************************************************************************************
00049 class ComunicacionSincronica
00050 {
00051     protected:
00052         const   Pin*     m_scl;
00053
00054     public:
00055         ComunicacionSincronica() = default;
00056         virtual void Write ( uint8_t data) = 0;
00057 //      virtual uint8_t* Read ( const uint8_t data ) = 0;           /**< Funcion de lectura */
00058         virtual ~ComunicacionSincronica() = default;
00059 };
00060
00061 #endif /* COMUNICACIONSINCRONICA_H_ */
```

7.87 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/14-ComSincronica/I2C/I2C.cpp File Reference

Descripcion del modulo.

`#include <Perifericos/14-ComSincronica/I2C/I2C.h>`

Include dependency graph for I2C.cpp:

Macros

- #define MAX_IC2 4

Variables

- I2C * g_i2c [MAX_IC2]

7.87.1 Detailed Description

Descripcion del modulo.

Date

5 oct. 2022

Author

Martinez Agustin

7.87.2 Macro Definition Documentation

7.87.2.1 MAX_IC2

```
#define MAX_IC2 4
```

Cantidad maxima de I2C que presenta el microcontrolador.

7.87.3 Variable Documentation

7.87.3.1 g_i2c

```
I2C* g_i2c[MAX_IC2]
```

Vector de UART s

7.88 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/14-ComSincronica/I2C/I2C.h File Reference

Breve descripción del objetivo del Módulo.

```
#include <Perifericos/01-Pin/Pin.h>
#include "LPC845.h"
#include <Perifericos/14-ComSincronica/ComunicacionSincronica.h>
```

Include dependency graph for I2C.h: This graph shows which files directly or indirectly include this file:

Classes

- class I2C

  Clase del objeto I2C El objeto I2C genera una comunicación sincrónica de tipo I2C. Posee las funciones basicas como start, stop, write y read.

Macros

- #define I2C_MAX_FREQ 400

7.88.1 Detailed Description

Breve descripción del objetivo del Módulo.

Date

5 oct. 2022

Author

Técnico Martinez Agustin

7.88.2 Macro Definition Documentation

7.88.2.1 I2C_MAX_FREQ

```
#define I2C_MAX_FREQ 400
```

Maxima frecuencia del I2C. Solo valida para I2C1 a I2C3

7.89 I2C.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************
00009
00036
00037
      /*****************************************************************************************************************
00038  *** MODULO
00039
      *****************************************************************************************************************
00040 #ifndef I2C_H_
00041 #define I2C_H_
00042
00046
00047
      /*****************************************************************************************************************
00048  *** INCLUDES GLOBALES
00049
      *****************************************************************************************************************
00050 #include <Perifericos/01-Pin/Pin.h>
00051 #include "LPC845.h"
00052 #include <Perifericos/14-ComSincronica/ComunicacionSincronica.h>
00053
      /*****************************************************************************************************************
00054  *** DEFINES GLOBALES
00055
      *****************************************************************************************************************
00056 #if defined (__cplusplus)
00057     extern "C" {
00058     void I2C0_IRQHandler ( void );
00059     }
00060     extern "C" {
00061     void I2C1_IRQHandler ( void );
00062     }
00063     extern "C" {
00064     void I2C2_IRQHandler ( void );
00065     }
00066     extern "C" {
00067     void I2C3_IRQHandler ( void );
00068     }
00069 #endif
00070
      /*****************************************************************************************************************
00071  *** MACROS GLOBALES
00072
      *****************************************************************************************************************
00073
00074
      /*****************************************************************************************************************
00075  *** TIPO DE DATOS GLOBALES
00076
      *****************************************************************************************************************
00077
00078
      /*****************************************************************************************************************
00079  *** VARIABLES GLOBALES
00080
      *****************************************************************************************************************
00081
00082
      /*****************************************************************************************************************
00083  *** IMPLANTACION DE UNA CLASE
00084
      *****************************************************************************************************************
00090
00091 class I2C: ComunicacionSincronica
00092 {       //La I2C1 a I2C3 no van a mas de 400khz
00093 public:
```

```
00095     #define     I2C_MAX_FREQ    400
00097     typedef enum {master = 1 , slave = 2} I2C_mode_t;
00099     typedef enum {write = 0 , read = 1 } I2C_action_t;
00104     typedef enum {idle = 0 , rx_data = 1 , tx_ready , NACK_addr , NACK_tx ,
00105                   slvst_addr = 0 , slvst_rx , slvst_tx ,
00106                   busy = 10} I2C_states_t;
00107
00108 private:
00109             I2C_Type*   m_I2C_register ;
00110     const   Pin*         m_sda;
00111             I2C_mode_t    m_mode ;
00112     const   uint8_t     m_slv_addr;
00113
00114     static  uint8_t     m_cant_created;
00115
00116 public:
00117             I2C             ( I2C_Type* I2C_register , Pin* sda , Pin* scl , I2C_mode_t mode =
      master,  uint8_t slv_addr = 0);
00118     void    Initialize        ( uint32_t clk_freq );
00119     void    EnableInterupt    ( void );
00120     void    DisableInterupt ( void );
00121
00122     void            Start      ( uint8_t addr , I2C_action_t action );
00123     void            Stop        ( void );
00124     void            Write      ( uint8_t data ) override;
00125     I2C&             operator=   ( uint8_t data );        //Sobrecarga de escritura
00126     int8_t          Read        ( uint8_t* data , bool continue_reading );
00127
00128     void            ACK          ( bool a );
00129     bool            ACKaddr      ( void );
00130     I2C_states_t    GetState    ( void );
00131
00132     virtual void    I2C_IRQHandler ( void ) { }
00133     virtual         ~I2C();
00134
00135 private:
00136     void    config          ( uint8_t& register_number );
00137     void    EnableSWM       ( void );
00138     void    ConfigClock     ( uint8_t& clk_offset , uint8_t& rst_offset );
00139     void    configBaudRate  ( uint32_t clk_freq );
00140 };
00141
00142 #endif /* I2C_H_ */
```

7.90 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/↩ Perifericos/14-ComSincronica/I2C/I2CMaster.h File Reference

Breve descripción del objetivo del Módulo.
`#include <Perifericos/14-ComSincronica/I2C/I2C.h>`
Include dependency graph for I2CMaster.h: This graph shows which files directly or indirectly include this file:

Classes

- class I2CMaster

  Clase del objeto I2CMaster El objeto I2CMaster genera una comunicación tipo master de I2C utilizando buffers de recepcion y transmision con interrupciones.

7.90.1 Detailed Description

Breve descripción del objetivo del Módulo.

Date

5 oct. 2022

Author

Técnico Martinez Agustin

7.91 I2CMaster.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00021
00022
      /*****************************************************************************************************************************
00023  *** MODULO
00024
      *****************************************************************************************************************************
00025 #ifndef I2CMASTER_H_
00026 #define I2CMASTER_H_
00027
00031
00032
      /*****************************************************************************************************************************
00033  *** INCLUDES GLOBALES
00034
      *****************************************************************************************************************************
00035 #include <Perifericos/14-ComSincronica/I2C/I2C.h>
00036
      /*****************************************************************************************************************************
00037  *** DEFINES GLOBALES
00038
      *****************************************************************************************************************************
00039
00040
      /*****************************************************************************************************************************
00041  *** MACROS GLOBALES
00042
      *****************************************************************************************************************************
00043
00044
      /*****************************************************************************************************************************
00045  *** TIPO DE DATOS GLOBALES
00046
      *****************************************************************************************************************************
00047
00048
      /*****************************************************************************************************************************
00049  *** VARIABLES GLOBALES
00050
      *****************************************************************************************************************************
00051
00052
      /*****************************************************************************************************************************
00053  *** IMPLANTACION DE UNA CLASE
00054
      *****************************************************************************************************************************
00060 class I2CMaster : protected I2C
00061 {
00062 private:
00063     uint8_t*    m_bufferRX;
00064     uint32_t    m_idxRxIn , m_idxRxOut , m_maxRx;
00065     uint8_t*    m_bufferTX;
00066     uint32_t    m_idxTxIn , m_idxTxOut , m_maxTx;
00067     uint32_t    m_cant_rw;
00068
00069     I2C_action_t m_action;
00070
00071 public:
00072             I2CMaster       (I2C_Type* I2C_register , Pin* sda , Pin* scl , uint32_t maxRx = 15,
      uint32_t maxTx = 15);
00073     void    Initialize        ( uint32_t clk_freq );
00074     void    Write           ( uint8_t addr, const char * msg );
00075     void    Write           ( uint8_t addr, const void * msg , uint32_t n );
00076     void    RequestRead       ( uint8_t addr, uint32_t cant_read = 1 );
00077     void*   Read            ( void * msg , uint32_t n );
00078     bool    isIdle            ( void );
00079     virtual ~I2CMaster();
00080
00081 private:
00082     void    I2C_IRQHandler  ( void ) override;
00083     void    pushRx          ( uint8_t dato );
00084     uint8_t popRx           ( uint8_t * dato );
00085     void    pushTx          ( uint8_t dato );
00086     uint8_t popTx           ( uint8_t * dato );
00087 };
00088
00089 #endif /* I2CMASTER_H_ */
```

7.92 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Drivers/Perifericos/14-ComSincronica/SPI/SPI.cpp File Reference

Descripcion del modulo.

`#include <Perifericos/14-ComSincronica/SPI/SPI.h>`

Include dependency graph for SPI.cpp:

Macros

- #define MAX_SPI 2

Variables

- SPI * g_spi [MAX_SPI]

7.92.1 Detailed Description

Descripcion del modulo.

Date

5 oct. 2022

Author

Martinez Agustin

7.92.2 Macro Definition Documentation

7.92.2.1 MAX_SPI

```
#define MAX_SPI 2
```

Cantidad maxima de I2C que presenta el microcontrolador.

7.92.3 Variable Documentation

7.92.3.1 g_spi

```
SPI* g_spi[MAX_SPI]
```

Vector de UART s

7.93 SPI.h

```
00001
      /*****************************************************************************************************************************
00009
00036
00037
      /*****************************************************************************************************************************
00038  *** MODULO
00039
      *****************************************************************************************************************************
00040 #ifndef I2C_H_
00041 #define I2C_H_
00042
00046
00047
      /*****************************************************************************************************************************
00048  *** INCLUDES GLOBALES
00049
      *****************************************************************************************************************************
00050 #include <Perifericos/01-Pin/Pin.h>
00051 #include "LPC845.h"
00052 #include <Perifericos/14-ComSincronica/ComunicacionSincronica.h>
00053
      /*****************************************************************************************************************************
00054  *** DEFINES GLOBALES
```

```
00055
      ***************************************************************************************************************************************
00056 #if defined (__cplusplus)
00057     extern "C" {
00058     void SPI0_IRQHandler ( void );
00059     }
00060     extern "C" {
00061     void SPI1_IRQHandler ( void );
00062     }
00063 #endif
00064
      /**************************************************************************************************************************************
00065  *** MACROS GLOBALES
00066
      ***************************************************************************************************************************************
00067
00068
      /**************************************************************************************************************************************
00069  *** TIPO DE DATOS GLOBALES
00070
      ***************************************************************************************************************************************
00071
00072
      /**************************************************************************************************************************************
00073  *** VARIABLES GLOBALES
00074
      ***************************************************************************************************************************************
00075
00076
      /**************************************************************************************************************************************
00077  *** IMPLANTACION DE UNA CLASE
00078
      ***************************************************************************************************************************************
00084
00085 class SPI: ComunicacionSincronica
00086 {       //La I2C1 a I2C3 no van a mas de 400khz
00087 public:
00088
00089     typedef enum {master = 1 , slave = 2} SPI_mode_t;
00090
00091 private:
00092
00093 public:
00094             SPI              ( );
00095     void    Initialize      ( uint32_t clk_freq );
00096     void    EnableInterupt  ( void );
00097     void    DisableInterupt ( void );
00098
00099     void            Write      ( uint8_t data );
00100     int8_t          Read        ( uint8_t* data , bool continue_reading );
00101
00102
00103     virtual void    SPI_IRQHandler ( void ) { }
00104     virtual         ~SPI();
00105
00106 private:
00107     void    config          ( uint8_t& register_number );
00108     void    EnableSWM       ( void );
00109     void    ConfigClock     ( uint8_t& clk_offset , uint8_t& rst_offset );
00110     void    configBaudRate  ( uint32_t clk_freq );
00111 };
00112
00113 #endif /* I2C_H_ */
```

7.94 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/↩ Firmware/dr_pll.cpp File Reference

Descripcion del modulo.

```
#include "dr_pll.h"
#include <LPC845.h>
```

Include dependency graph for dr_pll.cpp:

Functions

- void Inicializar_PLL (void)

  Inicializa el PLL en FREQ_PRINCIPAL MHz,. Por default el FRO interno tiene 12Mhz.

7.94.1 Detailed Description

Descripcion del modulo.

Date

5 feb. 2020

Author

Ing. Marcelo Trujillo

7.94.2 Function Documentation

7.94.2.1 Inicializar_PLL()

```
void Inicializar_PLL (
            void )
```

Inicializa el PLL en FREQ_PRINCIPAL MHz,. Por default el FRO interno tiene 12Mhz.

Author

Ing. Gustavo Fresno

Date

5 feb. 2020

7.95 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/Firmware/dr_pll.h File Reference

Breve descripción del objetivo del Módulo.
`#include <LPC845.h>`
Include dependency graph for dr_pll.h: This graph shows which files directly or indirectly include this file:

Functions

- void Inicializar_PLL (void)

  Inicializa el PLL en FREQ_PRINCIPAL MHz,. Por default el FRO interno tiene 12Mhz.

7.95.1 Detailed Description

Breve descripción del objetivo del Módulo.

Date

5 feb. 2020

Author

Ing. Marcelo Trujillo

7.95.2 Function Documentation

7.95.2.1 Inicializar_PLL()

```
void Inicializar_PLL (
            void )
```

Inicializa el PLL en FREQ_PRINCIPAL MHz,. Por default el FRO interno tiene 12Mhz.

Author

Ing. Gustavo Fresno

Date

5 feb. 2020

7.96 dr_pll.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013
00014 #ifndef DRIVERS_DR_PLL_H_
00015 #define DRIVERS_DR_PLL_H_
00016
00017
      /*****************************************************************************************************************************
00018  *** INCLUDES GLOBALES
00019
      *****************************************************************************************************************************
00020 #include <LPC845.h>
00021
      /*****************************************************************************************************************************
00022  *** DEFINES GLOBALES
00023
      *****************************************************************************************************************************
00024
00025
      /*****************************************************************************************************************************
00026  *** MACROS GLOBALES
00027
      *****************************************************************************************************************************
00028
00029
      /*****************************************************************************************************************************
00030  *** TIPO DE DATOS GLOBALES
00031
      *****************************************************************************************************************************
00032
00033
      /*****************************************************************************************************************************
00034  *** VARIABLES GLOBALES
00035
      *****************************************************************************************************************************
00036
00037
      /*****************************************************************************************************************************
00038  *** PROTOTIPOS DE FUNCIONES GLOBALES
00039
      *****************************************************************************************************************************
00040 void Inicializar_PLL( void ) ;
00041
00042 #endif /* DRIVERS_DR_PLL_H_ */
```

7.97 E:/Agustin/Escritorio/MCUexpresso/KitLPC845-C++/src/↩ Firmware/systick.h File Reference

Firmware del systick.

```
#include <LPC845.h>
#include <tipos.h>
#include <Perifericos/06-Callback/Callback.h>
#include <vector>
```

Include dependency graph for systick.h: This graph shows which files directly or indirectly include this file:

Functions

- uint32_t Inicializar_SysTick (uint32_t freq)

  Inicializa el systick en la frecuencia asignada.

Variables

- uint32_t g_systick_freq

7.97.1 Detailed Description

Firmware del systick.

Date

20 abr. 2022

Author

Marcelo

7.97.2 Function Documentation

7.97.2.1 Inicializar_SysTick()

```
uint32_t Inicializar_SysTick (
            uint32_t freq)
```

Inicializa el systick en la frecuencia asignada.

Parameters

in	*freq*	ticks por segundo

Returns

mensaje de error

7.97.3 Variable Documentation

7.97.3.1 g_systick_freq

```
uint32_t g_systick_freq  [extern]
```

Variable global. Posee la frecuencia del systick configurada.

7.98 systick.h

Go to the documentation of this file.

```
00001
      /*****************************************************************************************************************************
00009
00010
      /*****************************************************************************************************************************
00011  *** MODULO
00012
      *****************************************************************************************************************************
00013 #ifndef SYSTICK_H_
00014 #define SYSTICK_H_
00015
00016
      /*****************************************************************************************************************************
00017  *** INCLUDES GLOBALES
00018
      *****************************************************************************************************************************
00019 #include <LPC845.h>
00020 #include <tipos.h>
00021 #include <Perifericos/06-Callback/Callback.h>
00022 #include <vector>
00023 using namespace std;
00024
      /*****************************************************************************************************************************
00025  *** DEFINES GLOBALES
00026
      *****************************************************************************************************************************
```

```
00027
00028
      /*****************************************************************************************************************************
00029  *** MACROS GLOBALES
00030
      *****************************************************************************************************************************
00031
00032
      /*****************************************************************************************************************************
00033  *** TIPO DE DATOS GLOBALES
00034
      *****************************************************************************************************************************
00035 extern uint32_t g_systick_freq;
00036
      /*****************************************************************************************************************************
00037  *** IMPLANTACION DE LA CLASE
00038
      *****************************************************************************************************************************
00039 uint32_t Inicializar_SysTick( uint32_t freq );
00040
00041 #if defined (__cplusplus)
00042     extern "C" {
00043     void SysTick_Handler(void);
00044     }
00045 #endif
00046
00047 #endif /* SYSTICK_H_ */
```

Index