LPC845-C++

Generated by Doxygen 1.8.16

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

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

Module Documentation

5.1 Drivers

Classes

- class Pin

  Clase del objeto Pin.
- class gpio

  Clase del objeto gpio.
- class Output

  Clase del objeto outputs.
- class Input

  Clase del objeto Input.
- class swhandler

  Clase del objeto swhandler.
- 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 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 Uart

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

- 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 L298N

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

- 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 Reloj

  Clase del objeto Reloj.

- class Pwm

  Clase del objeto Pwm.

- class PWM_Reader

  Clase del objeto PWM_Reader.

- class Pin_interrupt

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

- 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 SCtimer

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

- 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.

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
- vector< swhandler ∗ > g_Handler
- std::vector< MRThandler ∗ > g_MRThandler
- Pin_interrupt ∗ g_gpiohandler [MAX_PININTERRUPT]

5.1.1 Detailed Description

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

```
Pin_interrupt* g_gpiohandler[MAX_PININTERRUPT]
```

Vector de interrupciones del PIN INTERRUPT

5.1.3.2 g_Handler

```
vector<swhandler*> g_Handler
```

Vector de interrupciones del systick

5.1.3.3 g_MRThandler

```
std::vector<MRThandler *> g_MRThandler
```

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 InOut

  Clase del objeto InOut.
- 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 ComunicacionAsincronica

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

5.2.1 Detailed Description

Chapter 6

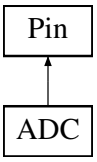
Class Documentation

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:



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.

Additional Inherited Members

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.

Returns

void

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.

Returns

void

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

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/19-ADC/ADC.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/19-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_I↩ NTEN** = 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.
- void Inicializar (void)
- 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.

6.2.5 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

Returns

void

6.2.5.2 EnableIrq()

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

Habilita la interrupcion.

Parameters

in	*irq*	Interrupcion a habilitar

Returns

void

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

Returns

void

6.2.5.5 Inicializar()

```
void ADC_Group::Inicializar (
            void  )
```

Destructor por defecto

6.2.5.6 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

6.2.5.7 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

6.2.5.8 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

6.2.5.9 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

Returns

void

6.2.5.10 SetSampleRate()

```
void ADC_Group::SetSampleRate (
            void  )
```

Setea la frecuencia de muestreo.

Returns

void

6.2.5.11 TriggerStartSeqA()

```
void ADC_Group::TriggerStartSeqA (
            void  )
```

Inicia la conversión analógica - digital de la secuencia A.

Returns

void

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

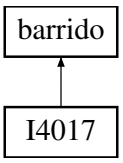
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/19-ADC/ADCGroup.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/19-ADC/ADCGroup.cpp

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:

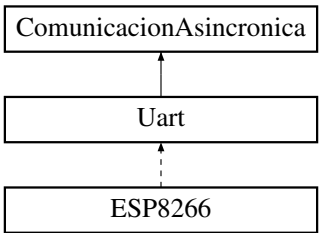
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08A-Barrido/Barrido.h

6.4 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

- 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

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

6.4.1 Detailed Description

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

6.4.2 Member Function Documentation

6.4.2.1 Message()

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

Funcion de lectura

Implemented in Uart, and ESP8266.

6.4.2.2 popRx()

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

Devuelve recepcion

6.4.2.3 popTx()

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

Devuelve transmision

6.4.2.4 pushRx()

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

Destructor por defecto Envia recepcion

6.4.2.5 pushTx()

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

Envia transmision

6.4.2.6 Transmit() [1/2]

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

Constructor por defecto Funcion de transmitir

Implemented in Uart, and ESP8266.

6.4.2.7 Transmit() [2/2]

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

Funcion de transmitir

Implemented in Uart, and ESP8266.

6.4.2.8 UART_IRQHandler()

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

Funcion de interrupcion

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

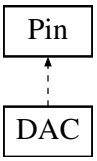
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/10-UART/ComunicacionAsincronica.h

6.5 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:



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

6.5.1 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.

6.5.2 Member Typedef Documentation

6.5.2.1 dac_channel

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

Canales del DAC

6.5.2.2 dac_error

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

Error en la clase DAC

6.5.3 Member Enumeration Documentation

6.5.3.1 dac_channel

```
enum DAC::dac_channel
```

Canales del DAC

6.5.3.2 dac_error

```
enum DAC::dac_error
```

Error en la clase DAC

6.5.4 Constructor & Destructor Documentation

6.5.4.1 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.

6.5.4.2 ~DAC()

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

Destructor por defecto

6.5.5 Member Function Documentation

6.5.5.1 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

6.5.5.2 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

6.5.5.3 Inicializar()

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

Inicializa el DAC.

Configura todos los registros de dicho DAC

Returns

Código de error

6.5.5.4 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

6.5.5.5 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

6.5.5.6 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

6.5.5.7 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

6.5.5.8 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

6.5.5.9 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

6.5.5.10 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

6.5.5.11 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

Returns

void

6.5.5.12 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

Returns

void

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

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/18-DAC/DAC.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/18-DAC/DAC.cpp

6.6 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.

6.6.1 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.

6.6.2 Member Enumeration Documentation

6.6.2.1 codigo_t

```
enum digito::codigo_t
```

Tipos de dígitos

6.6.2.2 modo_t

```
enum digito::modo_t
```

Modos de uso

6.6.2.3 SIMBOLOS

```
enum digito::SIMBOLOS
```

Simbolos a escribir

6.6.3 Constructor & Destructor Documentation

6.6.3.1 digito()

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

constructor por defecto

6.6.4 Member Function Documentation

6.6.4.1 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

Returns

void

6.6.4.2 Get()

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

Devuelve el valor del dígito.

Returns

valor del dígito.

6.6.4.3 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:

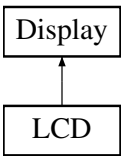
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08C-Display/Digito.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08C-Display/Digito.cpp

6.7 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

- virtual void Write (const int8_t *s)=0
- virtual void Clear (void)=0

6.7.1 Detailed Description

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

6.7.2 Member Function Documentation

6.7.2.1 Clear()

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

Funcion de limpieza genérica de un display

Implemented in LCD.

6.7.2.2 Write()

```
virtual void Display::Write (
            const int8_t * s )  [pure virtual]
```

Constructor por defecto Funcion de escritura genérica de un display

Implemented in LCD.

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

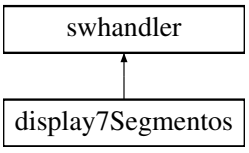
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/09-display_LCD/Display.h

6.8 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:



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.
- virtual ~display7Segmentos ()

6.8.1 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.

6.8.2 Constructor & Destructor Documentation

6.8.2.1 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.

6.8.2.2 ~display7Segmentos()

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

destructor por defecto

6.8.3 Member Function Documentation

6.8.3.1 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.

Returns

void

6.8.3.2 SWhandler()

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

Handler del display de 7 segmentos.

Funcion interrupción del display de 7 segmentos.

Returns

void

Implements swhandler.

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

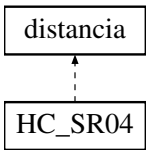
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08C-Display/Display7Segmentos
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08C-Display/Display7Segmentos

6.9 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

- virtual void Inicializar ()=0
- virtual uint32_t GetDistancia ()=0
- virtual bool operator== (uint32_t a)=0

6.9.1 Detailed Description

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

6.9.2 Member Function Documentation

6.9.2.1 GetDistancia()

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

Obtencion de distancia

Implemented in HC_SR04.

6.9.2.2 Inicializar()

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

Constructor por defecto Funcion inicializadora

Implemented in HC_SR04.

6.9.2.3 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:

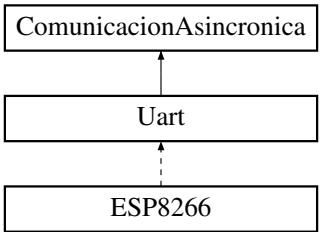
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/21-hc-sr04/distancia.h

6.10 ESP8266 Class Reference

Clase del objeto ESP8266 El objeto ESP8266 permite la simple utilización del módulo arduino ESP8266 y el E↩ SP01 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:



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

6.10.1 Detailed Description

Clase del objeto ESP8266 El objeto ESP8266 permite la simple utilización del módulo arduino ESP8266 y el E↩ SP01 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.

6.10.2 Member Enumeration Documentation

6.10.2.1 conection_type

```
enum ESP8266::conection_type
```

Tipo de conexión del cliente wifi

6.10.2.2 status_type

```
enum ESP8266::status_type
```

Enumeración con todos los estados posibles de la comunicación

6.10.3 Member Function Documentation

6.10.3.1 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

6.10.3.2 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

6.10.3.3 DisconnectToServer()

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

Se desconecta del server.

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

Returns

void

6.10.3.4 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.

Returns

void

6.10.3.5 GetIP()

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

Devuelve la IP del dispositivo.

Returns

int8_t∗: IP del dispositivo en formato string

6.10.3.6 GetStatus()

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

Devuelve el estado del dispositivo.

Returns

ESP8266::status_type: Estado del dispositivo.

6.10.3.7 Inicializar()

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

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

Configura los baudios y modo del dispositivo. FUNCION BLOQUEANTE

Returns

void

6.10.3.8 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.

6.10.3.9 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.

6.10.3.10 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.

Reimplemented from Uart.

6.10.3.11 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

Returns

void

6.10.3.12 Transmit() [1/2]

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

Sobrecarga de Transmit de UART.

Parameters

in	*msg*	Mensaje a enviar.

Returns

void

Reimplemented from Uart.

6.10.3.13 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.

Returns

void

Reimplemented from Uart.

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

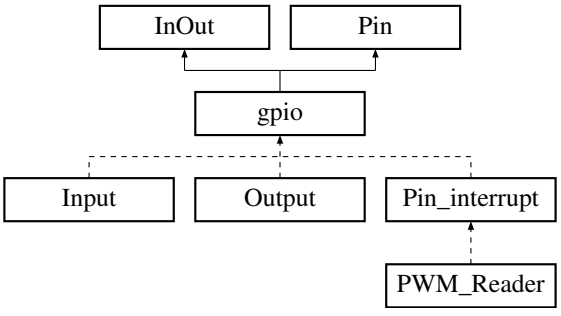
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/22-WiFi/ESP8266.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/22-WiFi/ESP8266.cpp

6.11 gpio Class Reference

Clase del objeto gpio.

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

Inheritance 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 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)

  Enciende la salida.
- uint8_t ClrPin (void)

  Limpia el pin.
- uint8_t SetTogglePin (void)

  Toggle del pin.
- uint8_t SetDir (void)

  Setea la direccion.
- uint8_t SetToggleDir (void)

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

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

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

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

  Sobrecarga del operador =.

Protected Attributes

- const mode_t m_mode
- direction_t m_direction
- const activity_t m_activity

Additional Inherited Members

6.11.1 Detailed Description

Clase del objeto gpio.

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

6.11.2 Constructor & Destructor Documentation

6.11.2.1 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

6.11.3 Member Function Documentation

6.11.3.1 ClrPin()

```
uint8_t gpio::ClrPin (
            void  )
```

Limpia el pin.

Coloca la salida con un 0 lógico

Returns

Devuelve el error

6.11.3.2 GetPin()

```
uint8_t gpio::GetPin (
            void  ) const
```

Devuelve el valor del pin.

Entrega el valor lógico de encendido o apagado de la entrada

Returns

Devuelve el error

6.11.3.3 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

6.11.3.4 SetDir()

```
uint8_t gpio::SetDir (
            void  )
```

Setea la direccion.

Configura la GPIO como entrada o como salida

Returns

Devuelve el error

6.11.3.5 SetPin()

```
uint8_t gpio::SetPin (
            void  )
```

Enciende la salida.

Coloca la salida con un 1 lógico

Returns

Devuelve el error

6.11.3.6 SetPinMode()

```
uint8_t gpio::SetPinMode (
            void  )
```

Configura el modo del pin.

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

Returns

Devuelve el error

6.11.3.7 SetPinResistor()

```
uint8_t gpio::SetPinResistor (
            void  )
```

Setea la resistencia interna.

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

Returns

Devuelve el error

6.11.3.8 SetToggleDir()

```
uint8_t gpio::SetToggleDir (
            void  )
```

Toggle de la dirección.

Pasa de output a input y viceversa

Returns

Devuelve el error

6.11.3.9 SetTogglePin()

```
uint8_t gpio::SetTogglePin (
            void  )
```

Toggle del pin.

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

Returns

Devuelve el error

6.11.4 Member Data Documentation

6.11.4.1 m_activity

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

Si es activo alto o activo bajo

6.11.4.2 m_direction

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

Si es input u output

6.11.4.3 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:

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/02-gpio/gpio.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/02-gpio/gpio.cpp

6.12 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

6.12.1 Detailed Description

Estructura de grupo de dígitos.

6.12.2 Constructor & Destructor Documentation

6.12.2.1 gruposdedigitos()

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

< Constructor por defecto

6.12.3 Member Data Documentation

6.12.3.1 m_cantidad

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

Cantidad de dígitos

6.12.3.2 m_comienzo

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

Comienzo del grupo

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

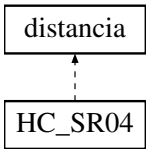
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08C-Display/GrupoDeDigitos.h

6.13 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:



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

6.13.1 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)

6.13.2 Constructor & Destructor Documentation

6.13.2.1 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.

6.13.3 Member Function Documentation

6.13.3.1 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.

6.13.3.2 Inicializar()

```
void HC_SR04::Inicializar (
            void  )  [virtual]
```

Inicializa todas las patas del ultrasónico.

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

Returns

void

Implements distancia.

6.13.3.3 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.

Returns

void

6.13.3.4 On()

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

Enciendo las patas del ultrasonico.

Llamo a las funciones de encender de tx y rx.

Returns

void

6.13.3.5 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

6.13.3.6 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

6.13.3.7 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.

6.13.3.8 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

6.13.3.9 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:

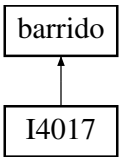
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/21-hc-sr04/HCSR04.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/21-hc-sr04/HCSR04.cpp

6.14 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:



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.

6.14.1 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.

6.14.2 Constructor & Destructor Documentation

6.14.2.1 I4017()

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

constructor por defecto

6.14.3 Member Function Documentation

6.14.3.1 Inicializar()

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

Función de Inicialización del I4017.

Returns

void

Implements barrido.

6.14.3.2 SetClock()

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

Enciende la pata clock.

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

Returns

void

6.14.3.3 SetDigito()

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

Función de cambio en el barrido.

Se mueve 1 en el barrido de bits.

Returns

void

Implements barrido.

6.14.3.4 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.

Returns

void

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

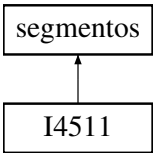
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08A-Barrido/I4017.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08A-Barrido/I4017.cpp

6.15 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:



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.

6.15.1 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.

6.15.2 Constructor & Destructor Documentation

6.15.2.1 I4511()

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

constructor por defecto

6.15.3 Member Function Documentation

6.15.3.1 Inicializar()

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

Función de Inicialización del I4511.

Returns

void

Implements segmentos.

6.15.3.2 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.

Returns

void

Implements segmentos.

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

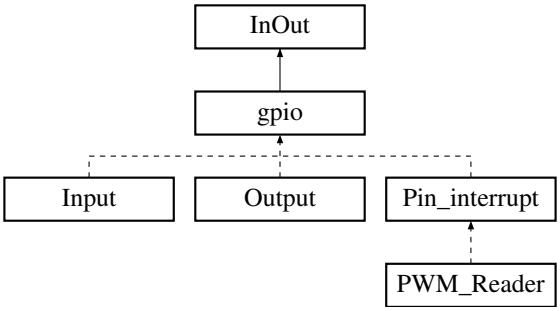
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08B-Segmentos/I4511.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08B-Segmentos/I4511.cpp

6.16 InOut Class Reference

Clase del objeto InOut.

`#include <InOut.h>`

Inheritance diagram for InOut:



Public Member Functions

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

6.16.1 Detailed Description

Clase del objeto InOut.

Clase abstracta pura para la generación de GPIO

6.16.2 Constructor & Destructor Documentation

6.16.2.1 ∼InOut()

```
virtual InOut::∼InOut ( )  [inline], [virtual]
```

Setea resistencia del pin

6.16.3 Member Function Documentation

6.16.3.1 ClrPin()

```
uint8_t InOut::ClrPin (
            void  )  [inline]
```

Enciende el pin

6.16.3.2 GetPin()

```
uint8_t InOut::GetPin (
            void  ) const  [inline]
```

Cambia la direccion del pin

6.16.3.3 SetDir()

```
uint8_t InOut::SetDir (
            void  )  [inline]
```

Cambia el valor del pin

6.16.3.4 SetPin()

```
uint8_t InOut::SetPin (
            void  )  [inline]
```

Constructor por defecto

6.16.3.5 SetPinMode()

```
uint8_t InOut::SetPinMode (
            void  )  [inline]
```

Obtiene valor del pin

6.16.3.6 SetPinResistor()

```
uint8_t InOut::SetPinResistor (
            void  )  [inline]
```

Setea modo del pin

6.16.3.7 SetToggleDir()

```
uint8_t InOut::SetToggleDir (
            void  )  [inline]
```

Setea la direccion del pin

6.16.3.8 SetTogglePin()

```
uint8_t InOut::SetTogglePin (
            void  )  [inline]
```

Apaga el pin

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

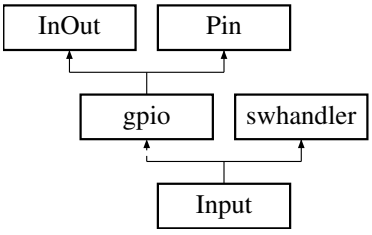
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/02-gpio/InOut.h

6.17 Input Class Reference

Clase del objeto Input.

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

Inheritance 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)

  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 ()

Friends

- bool operator== (uint32_t val, Input &I)

  Sobrecarga de del operador de asignacion.

Additional Inherited Members

6.17.1 Detailed Description

Clase del objeto Input.

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

6.17.2 Constructor & Destructor Documentation

6.17.2.1 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

6.17.2.2 ~Input()

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

Destructor por defecto

6.17.3 Member Function Documentation

6.17.3.1 get()

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

Devuelve el valor de la input.

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

Returns

void

6.17.3.2 Inicializar()

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

Inicializa la entrada.

Configura el GPIO y el buffer

Returns

void

6.17.3.3 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

6.17.3.4 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

6.17.3.5 SWhandler()

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

Funcion de interrupcion del systick.

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

Returns

void

Implements swhandler.

6.17.4 Friends And Related Function Documentation

6.17.4.1 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:

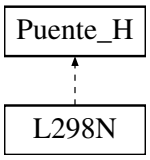
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/04-Input/Input.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/04-Input/Input.cpp

6.18 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:



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

6.18.1 Detailed Description

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

6.18.2 Constructor & Destructor Documentation

6.18.2.1 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.

6.18.2.2 ~L298N()

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

Destructor por defecto

6.18.3 Member Function Documentation

6.18.3.1 Avanzar()

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

Avanza los motores.

Enciende los 2 motores con tensión en la pata A.

Returns

void

Implements Puente_H.

6.18.3.2 Frenar()

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

Frena el motor.

Deja las cuatro salidas en 0.

Returns

void

Implements Puente_H.

6.18.3.3 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)

Returns

void

Implements Puente_H.

6.18.3.4 GirarDer()

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

Gira a la derecha.

Energiza solamente el motor 2 de forma directa.

Returns

void

Implements Puente_H.

6.18.3.5 GirarIzq()

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

Gira a la izquierda.

Energiza solamente el motor 1 de forma directa.

Returns

void

Implements Puente_H.

6.18.3.6 Inicializar()

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

Inicializa todas las salidas GPIO.

Setea la dirección y apaga todas las GPIO que posee.

Returns

void

Implements Puente_H.

6.18.3.7 Retroceder()

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

Pone los motores marcha atrás.

Enciende los motores en sentido contrario a "avanzar".

Returns

void

Implements Puente_H.

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

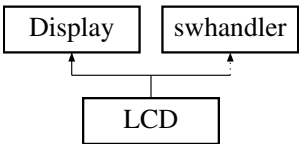
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/12-Puente_H/L298N.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/12-Puente_H/L298N.cpp

6.19 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:



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 int8_t *s)
  Escribe en el LCD.
- void Write (const int32_t n)
  Escribe en el LCD un número.
- LCD & operator= (const int8_t *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.

Protected Member Functions

- void SWhandler (void)

  Ejecuta la instrucción del LCD periodicamente.

6.19.1 Detailed Description

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

6.19.2 Member Enumeration Documentation

6.19.2.1 anonymous enum

```
anonymous enum
```

Posiciones del buffer

6.19.3 Constructor & Destructor Documentation

6.19.3.1 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.

6.19.4 Member Function Documentation

6.19.4.1 Clear()

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

Limpia el LCD.

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

Returns

void

Implements Display.

6.19.4.2 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.

Returns

void

6.19.4.3 operator=()

```
LCD & LCD::operator= (
            const int8_t * 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

6.19.4.4 SWhandler()

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

Ejecuta la instrucción del LCD periodicamente.

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

Returns

void

Implements swhandler.

6.19.4.5 Write() [1/2]

```
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.

Returns

void

6.19.4.6 Write() [2/2]

```
void LCD::Write (
            const int8_t * s )  [virtual]
```

Escribe en el LCD.

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

Parameters

in	*s*	String a escribir.

Returns

void

Implements Display.

6.19.4.7 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.

Returns

void

6.19.4.8 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.

Returns

void

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

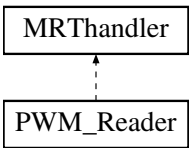
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/09-display_LCD/LCD.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/09-display_LCD/LCD.cpp

6.20 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 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.

6.20.1 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.

6.20.2 Constructor & Destructor Documentation

6.20.2.1 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.

6.20.3 Member Function Documentation

6.20.3.1 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

6.20.3.2 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

6.20.3.3 MRT_reset_time()

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

Resetea el timer.

Vuelve a cargar el tiempo máximo en el registro.

Returns

void

6.20.4 Member Data Documentation

6.20.4.1 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:

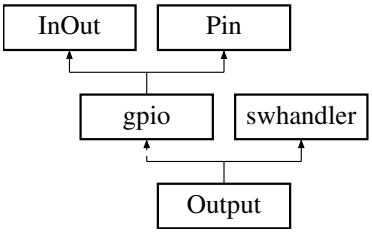
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/13-MultiRateTimer/MRThandler.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/13-MultiRateTimer/MRThandler.cpp

6.21 Output Class Reference

Clase del objeto outputs.

```
#include <Output.h>
```

Inheritance 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)

  Funcion de interrupcion del systick.
- virtual ~Output ()

Additional Inherited Members

6.21.1 Detailed Description

Clase del objeto outputs.

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

6.21.2 Constructor & Destructor Documentation

6.21.2.1 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

6.21.2.2 ~Output()

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

Destructor por defecto

6.21.3 Member Function Documentation

6.21.3.1 Off()

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

Funcion de apagado de la output.

Guarda en el buffer el estado de apagado

Returns

devuelve el error

6.21.3.2 On()

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

Funcion de encendido de la output.

Guarda en el buffer el estado de encendido

Returns

devuelve el error

6.21.3.3 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

6.21.3.4 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

6.21.3.5 SetUp()

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

Funcion de configuracion del output.

Setea la direccion y resistencia de la salida

Returns

devuelve el error

6.21.3.6 SWhandler()

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

Funcion de interrupcion del systick.

Coloca la salida con el valor del buffer

Returns

void

Implements swhandler.

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

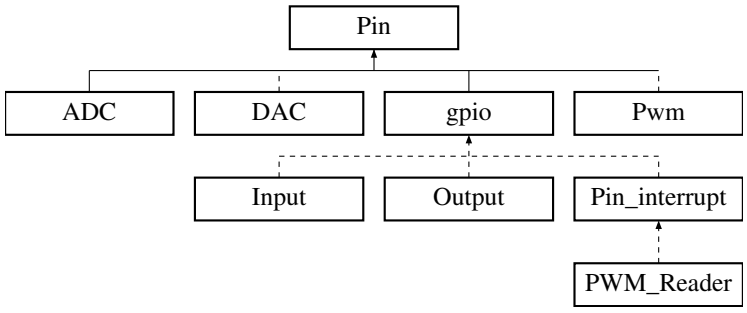
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/03-Output/Output.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/03-Output/Output.cpp

6.22 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

6.22.1 Detailed Description

Clase del objeto Pin.

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

6.22.2 Member Typedef Documentation

6.22.2.1 error_t

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

Enumeracion de error

6.22.2.2 port_t

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

Enumeracion con los puertos

6.22.3 Member Enumeration Documentation

6.22.3.1 error_t

```
enum Pin::error_t
```

Enumeracion de error

6.22.3.2 port_t

```
enum Pin::port_t
```

Enumeracion con los puertos

6.22.4 Constructor & Destructor Documentation

6.22.4.1 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

6.22.5 Member Data Documentation

6.22.5.1 m_bit

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

Guarda el bit del pin

6.22.5.2 m_error

```
int8_t Pin::m_error
```

Error en la clase pin

6.22.5.3 m_port

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

Guarda el puerto del pin

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

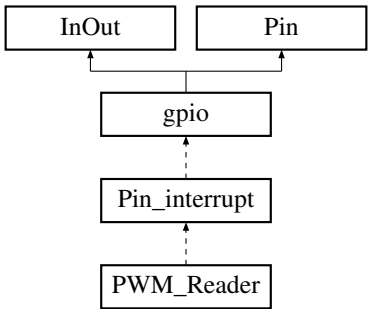
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/01-pin/Pin.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/01-pin/Pin.cpp

6.23 Pin_interrupt 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 Pin_interrupt:



Public Member Functions

- Pin_interrupt (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 ~Pin_interrupt ()

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

6.23.1 Detailed Description

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

6.23.2 Constructor & Destructor Documentation

6.23.2.1 Pin_interrupt()

```
Pin_interrupt::Pin_interrupt (
            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.

6.23.2.2 ~Pin_interrupt()

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

Destructor por defecto

6.23.3 Member Function Documentation

6.23.3.1 DisableInterupt()

```
void Pin_interrupt::DisableInterupt (
            void  )
```

Deshabilito la interrupción.

Deshabilito la interrupción de la pata específica que estoy utilizando.

Returns

void

6.23.3.2 EnableInterupt()

```
void Pin_interrupt::EnableInterupt (
            void  )
```

Habilito la interrupción.

Habilito la interrupción de la pata específica que estoy utilizando.

Returns

void

6.23.3.3 GpioHandler()

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

Funcion Handler que se ejecuta al realizarse la interrupcion

Implemented in PWM_Reader.

6.23.3.4 PinInterrupt_Inicializar()

```
void Pin_interrupt::PinInterrupt_Inicializar (
            void  )
```

Inicializo la interrupción por pin.

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

Returns

void

6.23.4 Member Data Documentation

6.23.4.1 m_cant

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

Cantidad de PIN INTERRUPT creados globalmente

6.23.4.2 m_interrput_mode

```
const uint8_t Pin_interrupt::m_interrput_mode
```

Tipo de interrupción del pin

6.23.4.3 m_interrupt_number

```
const uint8_t Pin_interrupt::m_interrupt_number
```

Número de PIN INTERRUPT

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

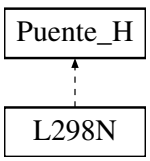
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/17-pin_interrupt/Pininterrupt.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/17-pin_interrupt/Pininterrupt.cpp

6.24 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

- 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

6.24.1 Detailed Description

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

6.24.2 Member Enumeration Documentation

6.24.2.1 anonymous enum

```
anonymous enum
```

Enumeracion de sentidos de giro

6.24.3 Member Function Documentation

6.24.3.1 Avanzar()

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

Avanza los motores

Implemented in L298N.

6.24.3.2 Frenar()

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

Frena los motores

Implemented in L298N.

6.24.3.3 Girar()

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

Gira los motores

Implemented in L298N.

6.24.3.4 GirarDer()

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

Gira a la derecha

Implemented in L298N.

6.24.3.5 GirarIzq()

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

Gira a la izquierda

Implemented in L298N.

6.24.3.6 Inicializar()

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

Constructor por defecto Inicializa el Puente H

Implemented in L298N.

6.24.3.7 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:

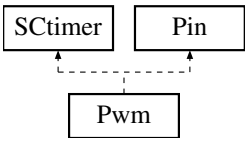
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/12-Puente_H/PuenteH.h

6.25 Pwm Class Reference

Clase del objeto Pwm.

`#include <Pwm.h>`

Inheritance 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

Additional Inherited Members

6.25.1 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.

6.25.2 Member Enumeration Documentation

6.25.2.1 activity_t

```
enum Pwm::activity_t
```

Enum de activo alto/bajo

6.25.2.2 pwm_channel_t

```
enum Pwm::pwm_channel_t
```

Canales del PWM

6.25.2.3 pwm_time_unit_t

```
enum Pwm::pwm_time_unit_t
```

Unidad de tiempo del PWM

6.25.3 Constructor & Destructor Documentation

6.25.3.1 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.

6.25.3.2 ~Pwm()

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

Destructor por defecto

6.25.4 Member Function Documentation

6.25.4.1 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

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

Returns

void

6.25.4.2 Off()

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

Apaga el PWM.

Utiliza los registros con StopTimer para deshabilitar la salida.

Returns

void

6.25.4.3 On()

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

Enciende el PWM.

Utiliza los registros con StarTimer para habilitar la salida.

Returns

void

6.25.4.4 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

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

Returns

void

6.25.4.5 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

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

Returns

void

6.25.5 Member Data Documentation

6.25.5.1 m_activity

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

Activo alto/bajo del PWM

6.25.5.2 m_pwm_channel

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

Canal del objeto PWM

6.25.5.3 m_toff

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

Tiempo de apagado del PWM en micro segundos

6.25.5.4 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:

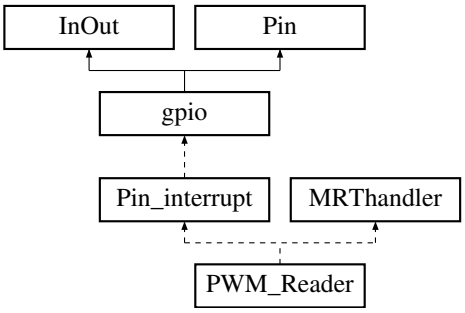
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/15-PWM/Pwm.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/15-PWM/Pwm.cpp

6.26 PWM_Reader Class Reference

Clase del objeto PWM_Reader.

`#include <PWMReader.h>`

Inheritance 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)

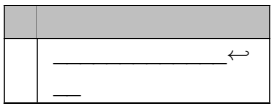
  Handler de la interrupción por flanco.

Additional Inherited Members

6.26.1 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----------—

6.26.2 Constructor & Destructor Documentation

6.26.2.1 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.

6.26.3 Member Function Documentation

6.26.3.1 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

6.26.3.2 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

Returns

void

Implements Pin_interrupt.

6.26.3.3 Inicializar()

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

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

Returns

void

6.26.3.4 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.

Returns

void

6.26.3.5 On()

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

Enciendo el PWM_In.
Habilito la interrupción.

Returns

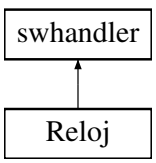
void

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

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/16-PWM_Reader/PWMReader.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/16-PWM_Reader/PWMReader.cpp

6.27 Reloj Class Reference

Clase del objeto Reloj.
`#include <Reloj.h>`
Inheritance 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.

Protected Member Functions

- void SWhandler (void)

  Actualiza el reloj.

6.27.1 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.

6.27.2 Constructor & Destructor Documentation

6.27.2.1 Reloj()

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

Constructor de clase Reloj.
Crea un Reloj por defecto

6.27.3 Member Function Documentation

6.27.3.1 GetHour()

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

Devuelve el valor de hora.
Entrega la variable m_hora.

Returns

m_hora

6.27.3.2 GetMin()

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

Devuelve el valor de minutos.
Entrega la variable m_minutos.

Returns

m_minutos

6.27.3.3 GetSeg()

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

Devuelve el valor de segundos.
Entrega la variable m_segundos.

Returns

m_segundos

6.27.3.4 Reset()

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

Resetea el reloj.
Coloca todas las variables en 0.

6.27.3.5 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.

6.27.3.6 SWhandler()

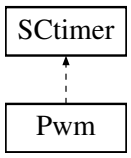
```
void Reloj::SWhandler (
            void  )  [protected], [virtual]
```

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

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/14-Reloj/Reloj.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/14-Reloj/Reloj.cpp

6.28 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 ()

6.28.1 Detailed Description

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

6.28.2 Constructor & Destructor Documentation

6.28.2.1 ~SCtimer()

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

Destructor por defecto

6.28.3 Member Function Documentation

6.28.3.1 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.

Returns

void

6.28.3.2 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.

Returns

void

6.28.3.3 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.

Returns

void

6.28.3.4 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.

Returns

void

6.28.3.5 StartTimer()

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

Habilita el timer.
Enciende el clock del timer.

Returns

void

6.28.3.6 StopTimer()

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

Deshabilita el timer.
Apaga el clock del timer.

Returns

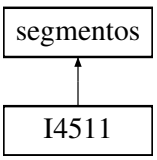
void

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

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/20-SCTimer/SCtimer.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/20-SCTimer/SCtimer.cpp

6.29 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

6.29.1 Detailed Description

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

6.29.2 Member Function Documentation

6.29.2.1 Inicializar()

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

Funcion de inicializacion
Implemented in I4511.

6.29.2.2 SetSegmentos()

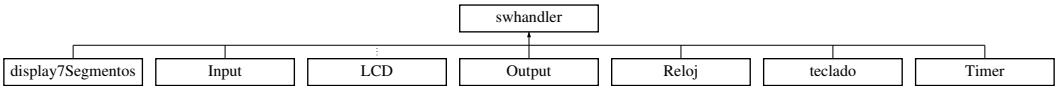
```
virtual void segmentos::SetSegmentos (
            uint16_t  )  [pure virtual]
```

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

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08B-Segmentos/Segmentos.h

6.30 swhandler Class Reference

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



Public Member Functions

- void SetInterrupt ()

  Activa la interrupción.
- void UnSetInterrupt ()

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

6.30.1 Detailed Description

Clase del objeto swhandler.
El objeto swhandler debe ser heredado y otorga a los herederos la conexión al systick y a la interrupción temporizada

6.30.2 Member Function Documentation

6.30.2.1 SetInterrupt()

```
void swhandler::SetInterrupt (
            void  )
```

Activa la interrupción.
Coloca al objeto dentro de la lista del handler systick

Returns

void

6.30.2.2 SWhandler()

```
virtual void swhandler::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 LCD, Timer, display7Segmentos, teclado, Reloj, Output, and Input.

6.30.2.3 UnSetInterrupt()

```
void swhandler::UnSetInterrupt (
            void  )
```

Desactiva la interrupción.
Borra de la lista handler al objeto actual

Returns

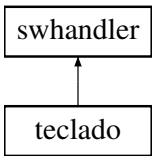
void

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

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/05-swhandler/swhandler.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/05-swhandler/swhandler.cpp

6.31 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:



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 ()

6.31.1 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.

6.31.2 Constructor & Destructor Documentation

6.31.2.1 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

6.31.2.2 ~teclado()

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

Destructor por defecto

6.31.3 Member Function Documentation

6.31.3.1 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

6.31.3.2 Inicializar()

```
void teclado::Inicializar (
            void  )
```

Inicializa el teclado.
Setea las direcciones y resistencias de las distintas entradas y salidas GPIO

Returns

void

6.31.3.3 SWhandler()

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

Funcion handler/interrupcion del teclado.
Ejecuta el barrido y la lectura del antirrebote

Returns

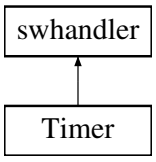
void

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

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/11-Teclado/teclado.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/11-Teclado/teclado.cpp

6.32 Timer Class Reference

Clase del objeto timer.
`#include <Timer.h>`
Inheritance 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.

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.

6.32.1 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.

6.32.2 Member Typedef Documentation

6.32.2.1 bases_t

```
typedef enum Timer::bases_t Timer::bases_t
```

Base de tiempo del timer

6.32.3 Member Enumeration Documentation

6.32.3.1 bases_t

```
enum Timer::bases_t
```
Base de tiempo del timer

6.32.4 Constructor & Destructor Documentation

6.32.4.1 Timer() [1/2]

```
Timer::Timer ( )
```
Constructor de clase timer.
Crea un timer por defecto

6.32.4.2 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

6.32.5 Member Function Documentation

6.32.5.1 ClrTmrEvent()

```
void Timer::ClrTmrEvent (
            void  )
```
Limpia el evento del timer.

Returns

void

6.32.5.2 GetmrStandBy()

```
bool Timer::GetmrStandBy (
            void  )
```
Informa si el timer está detenido o no.

Returns

True si el timer esta detenido. False si no.

6.32.5.3 GetTimer()

```
uint32_t Timer::GetTimer (
            void  ) const
```
Toma el valor al vuelo del timer en cuestion.
Lee el timer al vuelo

Returns

void

6.32.5.4 GetTmrEvent()

```
bool Timer::GetTmrEvent (
            void  )
```

Indica si el timer venció o no.

Returns

True si venció. False si no.

6.32.5.5 GetTmrRun()

```
uint32_t Timer::GetTmrRun (
            void  )
```

Obtiene el valor del timer.

Returns

Valor del timer.

6.32.5.6 operator bool()

```
Timer::operator bool ( )  [explicit]
```

Sobrecarga de del operador de contenido.

Returns

true por timer vencido y false por no vencido

6.32.5.7 operator"!()

```
bool Timer::operator!  ( )
```

Sobrecarga de del operador de negacion.

Returns

true por timer no vencido y false por vencido

6.32.5.8 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

6.32.5.9 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

6.32.5.10 SetmrStandBy()

```
void Timer::SetmrStandBy (
            uint8_t accion )
```

Detiene el timer sin apagarlo.

Parameters

in	*accion*	si lo detiene o no.

Returns

void.

6.32.5.11 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

Returns

void

6.32.5.12 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 tulizar

Returns

void

6.32.5.13 SetTmrEvent()

```
void Timer::SetTmrEvent (
            void  )
```

Setea el evento del timer.

Returns

void.

6.32.5.14 SetTmrHandler()

```
void Timer::SetTmrHandler (
            void  )
```

Ejecuta el timer Handler.
Llama a la función handler entregada por el constructor (siempe que no sea nullptr).

Returns

void

6.32.5.15 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

Returns

void

6.32.5.16 SWhandler()

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

Decremento periodico del timer. Debe ser llamada periodicamente con la base de tiempos.

Returns

void

Implements swhandler.

6.32.5.17 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

Returns

void

6.32.5.18 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

Returns

void

6.32.5.19 TimerStop()

```
void Timer::TimerStop (
            void  )
```

Detiene el timer.

Returns

void

6.32.5.20 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.

6.32.6 Friends And Related Function Documentation

6.32.6.1 operator==

```
bool Timer::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

6.32.7 Member Data Documentation

6.32.7.1 m_TmrBase

```
volatile bases_t Timer::m_TmrBase  [protected]
```

Base de tiempo del timer

6.32.7.2 m_TmrEvent

```
volatile bool Timer::m_TmrEvent  [protected]
```

True si el timer venció. False si el timer no venció

6.32.7.3 m_TmrHandler

```
void(* Timer::m_TmrHandler) (void)  [protected]
```

Función a ejecutar al terminar

6.32.7.4 m_TmrRun

```
volatile uint32_t Timer::m_TmrRun  [protected]
```

Tiempo de encendido del timer

6.32.7.5 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:

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/06-timer/Timer.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/06-timer/Timer.cpp

6.33 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.

6.33.1 Detailed Description

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

6.33.2 Constructor & Destructor Documentation

6.33.2.1 timers()

```
timers::timers ( )
```

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

6.33.3 Member Function Documentation

6.33.3.1 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

6.33.3.2 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

Returns

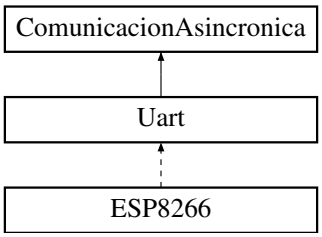
void

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

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/07-timers/Timers.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/07-timers/Timers.cpp

6.34 Uart Class Reference

Clase del objeto uart El objeto uart genera una comunicación asincrónica de tipo UART.
`#include <Uart.h>`
Inheritance 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)

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

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

  Modifica el baudrate al vuelo.

Additional Inherited Members

6.34.1 Detailed Description

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

6.34.2 Member Enumeration Documentation

6.34.2.1 bits_de_datos

`enum Uart::bits_de_datos`
Cantidad de bits de la UART

6.34.2.2 paridad_t

```
enum Uart::paridad_t
```
Tipo de paridad de la UART

6.34.3 Constructor & Destructor Documentation

6.34.3.1 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
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

6.34.4 Member Function Documentation

6.34.4.1 Message()

```
void * Uart::Message (
            void * msg,
            uint32_t n )  [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.
Reimplemented in ESP8266.

6.34.4.2 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

Returns

void

6.34.4.3 Transmit() [1/2]

```
void Uart::Transmit (
            const char * msg )  [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

Returns

void

Implements ComunicacionAsincronica.
Reimplemented in ESP8266.

6.34.4.4 Transmit() [2/2]

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

Funcion de transmitir
Implements ComunicacionAsincronica.
Reimplemented in ESP8266.
The documentation for this class was generated from the following files:

- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/10-UART/Uart.h
- C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/10-UART/Uart.cpp

Chapter 7

File Documentation

7.1 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/01-pin/Pin.h File Reference

Clase Abstracta de cualquier pin del microcontrolador.
`#include "tipos.h"`

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

7.1.1 Detailed Description

Clase Abstracta de cualquier pin del microcontrolador.

Date

10 ene. 2023

Author

Técnico Martinez Agustin

7.2 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/02-gpio/gpio.cpp File Reference

Descripcion del modulo.
`#include "gpio.h"`

7.2.1 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

7.3 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/02-gpio/gpio.h File Reference

Módulo con clase de manejo de GPIO.

```
#include <Drivers/01-pin/Pin.h>
#include <InOut.h>
#include <LPC845.h>
```

Classes

- class gpio

  Clase del objeto gpio.

7.3.1 Detailed Description

Módulo con clase de manejo de GPIO.

Date

22 jun. 2022

Author

Ing. Marcelo Trujillo

7.4 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/02-gpio/InOut.h File Reference

Clase Abstracta Pura de las GPIO.

```
#include <Drivers/01-pin/Pin.h>
```

Classes

- class InOut

  Clase del objeto InOut.

7.4.1 Detailed Description

Clase Abstracta Pura de las GPIO.

Date

22 jun. 2022

Author

Ing. Marcelo Trujillo

7.5 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/03-Output/Output.cpp File Reference

Clase del tipo Output o salida digital.

```
#include <Drivers/03-Output/Output.h>
```

7.5.1 Detailed Description

Clase del tipo Output o salida digital.

Date

12 jul. 2022

Author

Ing. Marcelo Trujillo

7.6 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/03-Output/Output.h File Reference

Clase del tipo Output o salida digital.

```
#include "gpio.h"
#include "swhandler.h"
```

Classes

- class Output

  Clase del objeto outputs.

7.6.1 Detailed Description

Clase del tipo Output o salida digital.

Date

12 jul. 2022

Author

Ing. Marcelo Trujillo

7.7 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/04-Input/Input.cpp File Reference

funciones miembro de la clase Input
`#include <Drivers/04-Input/Input.h>`

Functions

- bool operator== (uint32_t val, Input &I)

  Sobrecarga de del operador de asignacion.

7.7.1 Detailed Description

funciones miembro de la clase Input

Date

27 may. 2022

Author

Ing. Marcelo Trujillo

7.7.2 Function Documentation

7.7.2.1 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

7.8 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/04-Input/Input.h File Reference

funciones miembro de la clase Input
`#include "swhandler.h"`
`#include "gpio.h"`

Classes

- class Input

  Clase del objeto Input.

Macros

- #define MAX_BOUNCE 4

7.8.1 Detailed Description

funciones miembro de la clase Input

Date

27 may. 2022

Author

Ing. Marcelo Trujillo

7.8.2 Macro Definition Documentation

7.8.2.1 MAX_BOUNCE

```
#define MAX_BOUNCE 4
```

Cantidad de rebotes por defecto

7.9 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/05-swhandler/swhandler.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>
```

Classes

- class swhandler

  Clase del objeto swhandler.

Macros

- #define TICK_SECONDS(x) ((x) * g_systick_freq)
- #define TICK_MILISECONDS(x) ((x) * (g_systick_freq/1000))
- #define TICK_MICROSECONDS(x) ((x) * (g_systick_freq/1000000))

Variables

- vector< swhandler * > g_Handler

7.9.1 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

7.9.2 Macro Definition Documentation

7.9.2.1 TICK_MICROSECONDS

```
#define TICK_MICROSECONDS(
            x ) ((x) * (g_systick_freq/1000000))
```

MACRO expresion to obtain the corresponding TICKS for "x" microseconds

7.9.2.2 TICK_MILISECONDS

```
#define TICK_MILISECONDS(
            x ) ((x) * (g_systick_freq/1000))
```

MACRO expresion to obtain the corresponding TICKS for "x" miliseconds

7.9.2.3 TICK_SECONDS

```
#define TICK_SECONDS(
            x ) ((x) * g_systick_freq)
```

MACRO expresion to obtain the corresponding TICKS for "x" Seconds

7.10 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/06-timer/Timer.h File Reference

Clase para creacion de temporizadores.

```
#include "swhandler.h"
#include "tipos.h"
```

Classes

- class Timer

  Clase del objeto timer.

Typedefs

- typedef void(∗ Timer_Handler) (void)

7.10.1 Detailed Description

Clase para creacion de temporizadores.

Date

4 may. 2022

Author

Ing. Marcelo Trujillo

7.11 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/07-timers/Timers.cpp File Reference

funciones miembro de la clase timers

```
#include <Timers.h>
```

7.11.1 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

7.12 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/07-timers/Timers.h File Reference

Breve descripción del objetivo del Módulo.
`#include <timer.h>`

Classes

- class timers

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

7.12.1 Detailed Description

Breve descripción del objetivo del Módulo.

Date

10 jul. 2022

Author

Ing. Marcelo Trujillo

7.13 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/08-display_segmentos/08A-Barrido/Barrido.h File Reference

Clase abstracta de manejo barridos de datos.

Classes

- class barrido

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

7.13.1 Detailed Description

Clase abstracta de manejo barridos de datos.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.14 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/08-display_segmentos/08A-Barrido/I4017.cpp File Reference

Objeto de control del integrado I4017.

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

7.14.1 Detailed Description

Objeto de control del integrado I4017.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.15 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/08-display_segmentos/08A-Barrido/I4017.h File Reference

Objeto de control del integrado I4017.

```
#include <barrido.h>
#include "gpio.h"
#include <vector>
```

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.

7.15.1 Detailed Description

Objeto de control del integrado I4017.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.16 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08B-Segmentos/I4511.cpp File Reference

Objeto de control del integrado I4511.

```
#include <I4511.h>
```

7.16.1 Detailed Description

Objeto de control del integrado I4511.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.17 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08B-Segmentos/I4511.h File Reference

Objeto de control del integrado I4511.

```
#include <segmentos.h>
#include "gpio.h"
#include <vector>
```

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.

7.17.1 Detailed Description

Objeto de control del integrado I4511.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.18 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08B-Segmentos/Segmentos.h File Reference

Clase abstracta de manejo de segmentos binarios.

```
#include "tipos.h"
```

Classes

- class segmentos

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

7.18.1 Detailed Description

Clase abstracta de manejo de segmentos binarios.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.19 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/08-display_segmentos/08C-Display/Digito.cpp File Reference

Objeto dígito genérico para implementaciones posteriores.
`#include <digito.h>`

Variables

- const uint8_t Tabla_Digitos_BCD_7seg [ ] = {CERO,UNO,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,OC↩ HO,NUEVE}

7.19.1 Detailed Description

Objeto dígito genérico para implementaciones posteriores.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

7.19.2 Variable Documentation

7.19.2.1 Tabla_Digitos_BCD_7seg

```
const uint8_t Tabla_Digitos_BCD_7seg[ ] = {CERO,UNO,DOS,TRES,CUATRO,CINCO,SEIS,SIETE,OCHO,NUE↩
VE}
```

Taba de dígitos BCD en binario

7.20 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/08-display_segmentos/08C-Display/Digito.h File Reference

Objeto dígito genérico para implementaciones posteriores.
`#include "tipos.h"`

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.

7.20.1 Detailed Description

Objeto dígito genérico para implementaciones posteriores.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

7.21 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08C-Display/Display7Segmentos.cpp File Reference

Clase para la creación de displays de 7 segmentos.

```
#include <display7Segmentos.h>
```

7.21.1 Detailed Description

Clase para la creación de displays de 7 segmentos.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

7.22 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/08-display_segmentos/08C-Display/Display7Segmentos.h File Reference

Clase para la creación de displays de 7 segmentos.

```
#include <barrido.h>
#include <segmentos.h>
#include <digito.h>
#include <grupodedigitos.h>
#include "gpio.h"
#include "swhandler.h"
#include <vector>
```

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)

7.22.1 Detailed Description

Clase para la creación de displays de 7 segmentos.

Date

26 jul. 2022

Author

Ing. Marcelo Trujillo

7.22.2 Macro Definition Documentation

7.22.2.1 UPDATE_TICKS

```
#define UPDATE_TICKS (3)
```

Velocidad de escritura de los leds. MODIFICAR EN CASO DE SER NECESARIO

7.23 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/08-display_segmentos/08C-Display/GrupoDeDigitos.h File Reference

Clase para la agrupación de dígitos.

```
#include <digito.h>
#include "tipos.h"
```

Classes

- struct gruposdedigitos

  Estructura de grupo de dígitos.

7.23.1 Detailed Description

Clase para la agrupación de dígitos.

Date

27 jul. 2022

Author

Ing. Marcelo Trujillo

7.24 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/09-display_LCD/Display.h File Reference

Clase base para objetos del tipo pantallas/displays.

```
#include "tipos.h"
```

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 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/09-display_LCD/LCD.h File Reference

Clase para un LCD con comunicación de 4 patas, solo escritura.

```
#include <Display.h>
#include <swhandler.h>
#include <gpio.h>
#include <vector>
```

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*/

7.25.1 Detailed Description

Clase para un LCD con comunicación de 4 patas, solo escritura.

Date

22 jun. 2022

Author

Técnico. Martinez Agustin

7.25.2 Macro Definition Documentation

7.25.2.1 CLEAR_DISPLAY

```
#define CLEAR_DISPLAY 0b00000001
```
Instruccion de limpieza del display

7.25.2.2 CURSOR_DISPLAY_SHIFT

```
#define CURSOR_DISPLAY_SHIFT 0b00010000 /** 1 S/C R/L - -*/
```
Instruccion de cursor del display

7.25.2.3 DISPLAY_CONTROL

```
#define DISPLAY_CONTROL 0b00001000 /** 1 D C B*/
```
Instruccion de control del display

7.25.2.4 ENTRY_MODE_SET

```
#define ENTRY_MODE_SET 0b00000100 /** 1 I/D S*/
```
Instruccion de entrada de escritura del display

7.25.2.5 FUNCTION_SET

```
#define FUNCTION_SET 0b00100000 /** 1 DL N F - -*/
```
Instruccion de modo de funcionamiento del display

7.25.2.6 RETURN_HOME

```
#define RETURN_HOME 0b00000010
```
Instruccion de regreso a posición 0 del display

7.25.2.7 SET_CGRAM

```
#define SET_CGRAM 0b01000000 /** 1 ACG ACG ACG ACG ACG ACG*/
```
Instruccion de seteo de CGRAM del display

7.25.2.8 SET_DDRAM

```
#define SET_DDRAM 0b10000000 /** 1 ADD ADD ADD ADD ADD ADD ADD*/
```
Instruccion de seteo de DDRAM del display

7.26 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/10-UART/ComunicacionAsincronica.h File Reference

Objeto base para la creacion de comunicaciones asincrónicas.

```
#include "tipos.h"
```

Classes

- class ComunicacionAsincronica

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

7.26.1 Detailed Description

Objeto base para la creacion de comunicaciones asincrónicas.

Date

5 oct. 2022

Author

Ing. Marcelo Trujillo

7.27 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/10-UART/Uart.cpp File Reference

Descripcion del modulo.

```
#include <Uart.h>
```

Variables

- ComunicacionAsincronica * g_usart [5]

7.27.1 Detailed Description

Descripcion del modulo.

Date

5 oct. 2022

Author

Ing. Marcelo Trujillo

7.27.2 Variable Documentation

7.27.2.1 g_usart

```
ComunicacionAsincronica* g_usart[5]
```

Vector de UART s

7.28 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/10-UART/Uart.h File Reference

Breve descripción del objetivo del Módulo.

```
#include <ComunicacionAsincronica.h>
#include "LPC845.h"
#include <Pin.h>
```

Classes

- class Uart

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

7.28.1 Detailed Description

Breve descripción del objetivo del Módulo.

Date

5 oct. 2022

Author

Ing. Marcelo Trujillo

7.29 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/11-Teclado/teclado.cpp File Reference

Descripcion del modulo.

```
#include <teclado.h>
```

7.29.1 Detailed Description

Descripcion del modulo.

Date

24 jul. 2022

Author

Ing. Marcelo Trujillo

7.30 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/11-Teclado/teclado.h File Reference

Breve descripción del objetivo del Módulo.

```
#include "gpio.h"
#include "swhandler.h"
#include <vector>
```

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

7.30.1 Detailed Description

Breve descripción del objetivo del Módulo.

Date

24 jul. 2022

Author

Ing. Marcelo Trujillo

7.30.2 Macro Definition Documentation

7.30.2.1 NO_KEY

```
#define NO_KEY 0xff
```

Número representativo a "ninguna tecla se a presionado"

7.31 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/12-Puente_H/L298N.h File Reference

Clase del módulo de puente H LN298N.

```
#include <Drivers/12-Puente_H/PuenteH.h>
#include "gpio.h"
```

Classes

- class L298N

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

7.31.1 Detailed Description

Clase del módulo de puente H LN298N.

Date

25 sep. 2022

Author

Técnico. Martinez Agustin

7.32 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/12-Puente_H/PuenteH.h File Reference

Clase base para objetos del tipo Puente H.

```
#include "tipos.h"
```

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.

7.32.1 Detailed Description

Clase base para objetos del tipo Puente H.

Date

25 sep. 2022

Author

Técnico. Martinez Agustin

7.33 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/13-MultiRateTimer/MRThandler.cpp File Reference

Handler del timer MRT.

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

Variables

- std::vector< MRThandler * > g_MRThandler

7.33.1 Detailed Description

Handler del timer MRT.

Date

2 sep. 2022

Author

Técnico Martinez Agustin

7.34 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/14-Reloj/Reloj.h File Reference

Objeto que guardará el tiempo desde que se creo.

```
#include "swhandler.h"
#include "tipos.h"
```

Classes

- class Reloj

  Clase del objeto Reloj.

7.34.1 Detailed Description

Objeto que guardará el tiempo desde que se creo.

Date

27 nov. 2022

Author

Técnico Martinez Agustin

7.35 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/15-PWM/Pwm.h File Reference

Generador de PWM sin interrupción.

```
#include <Drivers/20-SCTimer/SCtimer.h>
#include "Pin.h"
```

Classes

- class Pwm

  Clase del objeto Pwm.

7.35.1 Detailed Description

Generador de PWM sin interrupción.

Date

7 oct. 2022

Author

Técnico. Martinez Agustin

7.36 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/16-PWM_Reader/PWMReader.h File Reference

Pata que lee tamaños de pulsos de entrada.

```
#include <Drivers/17-pin_interrupt/Pininterrupt.h>
#include "MRThandler.h"
```

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 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/17-pin_interrupt/Pininterrupt.h File Reference

Clase para entradas con interrupciones por flanco.

```
#include <Drivers/01-pin/Pin.h>
#include "gpio.h"
#include <LPC845.h>
```

Classes

- class Pin_interrupt

  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

- Pin_interrupt * g_gpiohandler [MAX_PININTERRUPT]

7.37.1 Detailed Description

Clase para entradas con interrupciones por flanco.

Date

17 sep. 2022

Author

Grupo 4

7.37.2 Macro Definition Documentation

7.37.2.1 MAX_PININTERRUPT

```
#define MAX_PININTERRUPT 8
```

Cantidad máxima de interrupciones de pin

7.38 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/18-DAC/DAC.h File Reference

Objeto DAC.

```
#include "LPC845.h"
#include "tipos.h"
#include "Pin.h"
```

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)

7.38.1 Detailed Description

Objeto DAC.

Date

12 ene. 2023

Author

Técnico Martinez Agustin

7.38.2 Macro Definition Documentation

7.38.2.1 MAX_DAC_CHANNEL

```
#define MAX_DAC_CHANNEL (2)
```

Máximos canales del DAC

7.38.2.2 MAX_DAC_VALUE

```
#define MAX_DAC_VALUE (0x3FF)
```

Máximo valor del DAC por defecto

7.39 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/19-ADC/ADC.cpp File Reference

Breve descripcion del modulo.

```
#include "ADC.h"
```

Variables

- uint8_t **pin_index** [MAX_ADC_CHANNELS] = { 7 , 6 , 14 , 23 , 22 , 21 , 20 , 19 , 18 , 17 , 13 , 4 }

7.39.1 Detailed Description

Breve descripcion del modulo.

Date

22 feb. 2023

Author

Técnico Martinez Agustín

Version

v1.0

7.40 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/19-ADC/ADC.h File Reference

Breve descripción del Módulo.

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

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.

7.40.1 Detailed Description

Breve descripción del Módulo.

Date

22 feb. 2023

Author

Técnico Martinez Agustín

Version

v1.0

7.41 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/19-ADC/ADCGroup.cpp File Reference

Objeto DAC.

```
#include <ADCGroup.h>
```

Variables

- ADC_Group * g_adc = nullptr

  vector de interrupciones ADC

7.41.1 Detailed Description

Objeto DAC.

Date

13 nov. 2022

Author

Federico

7.42 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/20-SCTimer/SCtimer.cpp File Reference

Salida autónoma temporizada.

```
#include <Drivers/20-SCTimer/SCtimer.h>
```

7.42.1 Detailed Description

Salida autónoma temporizada.

Date

7 oct. 2022

Author

Grupo 4

7.43 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/20-SCTimer/SCtimer.h File Reference

Salida autónoma temporizada.

```
#include "LPC845.h"
```

Classes

- class SCtimer

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

7.43.1 Detailed Description

Salida autónoma temporizada.

Date

7 oct. 2022

Author

Grupo 4

7.44 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/21-hc-sr04/distancia.h File Reference

Clase base para objetos medidores de distancias.

```
#include "tipos.h"
```

Classes

- class distancia

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

7.44.1 Detailed Description

Clase base para objetos medidores de distancias.

Date

22 jun. 2022

Author

Técnico. Martinez Agustin

7.45 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/21-hc-sr04/HCSR04.h File Reference

Clase del sensor ultrasónico HCSR04.

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

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) (((a)∗164) / 10000)
- #define DISTANCIA_MAX 400

7.45.1 Detailed Description

Clase del sensor ultrasónico HCSR04.

Date

22 jun. 2022

Author

Técnico. Martinez Agustin

7.45.2 Macro Definition Documentation

7.45.2.1 CALC_DISTANCIA

```
#define CALC_DISTANCIA(
              a ) ( ((a)∗164) / 10000 )
```

Fórmula de cálculo tiempo->distancia

7.45.2.2 DISTANCIA_MAX

```
#define DISTANCIA_MAX 400
```

Distancia máxima en centimetros

7.45.2.3 PERIODO

```
#define PERIODO 80
```

Perido del PWM En milisegundos

7.46 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/↩ Drivers/22-WiFi/ESP8266.cpp File Reference

Breve descripcion del modulo.

```
#include "ESP8266.h"
```

7.46.1 Detailed Description

Breve descripcion del modulo.

Date

2 mar. 2023

Author

Técnico Martinez Agustín

Version

v1.0

7.47 C:/Users/mari-ser/Desktop/MCUexpresso/KitLPC845-C++/src/Drivers/22-WiFi/ESP8266.h File Reference

Breve descripción del Módulo.

```
#include <Uart.h>
#include <Timer.h>
```

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

7.47.1 Detailed Description

Breve descripción del Módulo.

Date

2 mar. 2023

Author

Técnico Martinez Agustín

Version

v1.0

7.47.2 Macro Definition Documentation

7.47.2.1 DEFAULT_ESP01_BAUDRATE

```
#define DEFAULT_ESP01_BAUDRATE 115200
```

Velocidad de transmision del ESP por defecto, antes de haber sido modificado

7.47.2.2 SEG_ESP01_TIMEOUT

```
#define SEG_ESP01_TIMEOUT 20
```

Tiempo por default de espera para la realización de los comandos AT