CML Microcircuits

COMMUNICATION SEMICONDUCTORS

# EC0003 C Driver Libraries PE0003_DriverLib

Publication: CL/PE0003/UM/1 July 2015

EC0003 C Code Development Environment
PE0003 Peripheral Library Control

## Contents

## 1 Introduction

This document provides a description of the API used for configuration and interaction with the PE0003 peripherals. It has been created using Doxygen – an automatic documentation generation tool used to produce software reference documents. Content is created from within the code itself and therefore offers intuitive cross referencing between the document and code and provides an easy path to future updating.

### 1.1 History

| Version | Changes | Date |
|---|---|---|
| 1 | First Release | 22 July 2015 |

# 2 Data Structure Index

## 2.1 Data Structures

The following are the data structures with brief descriptions:

# 3 File Index

## 3.1 File List

The following is a list of all files. Brief descriptions are given for each in the relevant sections:

# 4 Data Structure Documentation

## 4.1 PE0003_IOBit Struct Reference

Used to remap the pins.

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

### 4.1.1 Data Fields

- uint8_t **port**
- uint32_t **pin**

### 4.1.2 Detailed Description

Used to remap the pins.

### 4.1.3 Field Documentation

#### uint32_t PE0003_IOBit::pin

#### uint8_t PE0003_IOBit::port

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

- inc/**gpio.h**

## 4.2 inc/cbus.h File Reference

```
#include "chip.h"
```

### 4.2.1 Macros

- #define **CBUS_SPEED** 10000000

### 4.2.2 Functions

- void **Pe0003_CbusInit** (LPC_SSP_T *pSSP)
  *Initialise the C-BUS port, using the default configuration.*
- void **Pe0003_CbusDeInit** (LPC_SSP_T *pSSP)
  *Deinitialise the C-BUS port.*
- void **Pe0003_CbusRegisterWrite** (LPC_SSP_T *pSSP, uint8_t cbus_address, uint16_t *data_ptr, uint8_t bytes_per_access, uint8_t accesses)
  *Streaming write of bytes or words to a C-BUS port.*
- void **Pe0003_CbusRegisterRead** (LPC_SSP_T *pSSP, uint8_t cbus_address, uint16_t *data_ptr, uint8_t bytes_per_access, uint8_t accesses)
  *Streaming read of bytes or words from a C-BUS port.*
- void **Pe0003_CbusEnable** (LPC_SSP_T *pSSP)
  *Enable the C-BUS port.*
- void **Pe0003_CbusDisable** (LPC_SSP_T *pSSP)
  *Disable the C-BUS port.*
- uint8_t **Pe0003_CbusRxFifoEmpty** (LPC_SSP_T *pSSP)
  *Check empty FIFO status.*
- void **Pe0003_CbusSetFrequency** (LPC_SSP_T *pSSP, uint32_t frequency)
  *Change the C-BUS frequency from the 10MHz default.*
- void **Pe0003_CbusStart** (LPC_SSP_T *pSSP)
  *Start the C-BUS port.*
- void **Pe0003_CbusStop** (LPC_SSP_T *pSSP)
  *Stop the C-BUS port.*
- void **Pe0003_CbusSetSpeed** (LPC_SSP_T *pSSP, uint32_t iSpeed)
  *Set the speed of the C-BUS SCLK in Hz.*
- void **Pe0003_CbusWriteNoData** (LPC_SSP_T *pSSP, uint32_t iAddr)
  *C-BUS write no data to an specific address.*
- void **Pe0003_CbusGeneralReset** (LPC_SSP_T *pSSP)
  *C-BUS general reset.*
- void **Pe0003_CbusWriteByte** (LPC_SSP_T *pSSP, uint32_t iAddr, uint32_t iData)
  *Write a byte to a C-BUS port.*
- void **Pe0003_CbusWriteWord** (LPC_SSP_T *pSSP, uint32_t iAddr, uint32_t iData)
  *Write a word to a C-BUS port.*
- void **Pe0003_CbusWriteBstream** (LPC_SSP_T *pSSP, uint32_t iAddr, uint8_t *pData, uint32_t iCount)
  *Write a byte stream to a C-BUS port.*
- void **Pe0003_CbusWriteWstream** (LPC_SSP_T *pSSP, uint32_t iAddr, uint16_t *pData, uint32_t iCount)
  *Write a byte stream to a C-BUS port.*
- uint32_t **Pe0003_CbusReadByte** (LPC_SSP_T *pSSP, uint32_t iAddr)
  *Read a byte from a C-BUS port.*
- uint32_t **Pe0003_CbusReadWord** (LPC_SSP_T *pSSP, uint32_t iAddr)
  *Read a word from a C-BUS port.*
- void **Pe0003_CbusReadBstream** (LPC_SSP_T *pSSP, uint32_t iAddr, uint8_t *pData, uint32_t iCount)

*Read a byte from a C-BUS port.*

- void **Pe0003_CbusReadWstream** (LPC_SSP_T *pSSP, uint32_t iAddr, uint16_t *pData, uint32_t iCount)
  *Read a word from a C-BUS port.*

### 4.2.3 Macro Definition Documentation

**#define CBUS_SPEED  10000000**

### 4.2.4 Function Documentation

**void Pe0003_CbusDeInit (LPC_SSP_T *  *pSSP*)**

Deinitialise the C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1, CBUS2. |

**Returns:**

None

**Note:**

Legacy function - use Pe0003_CbusStop instead.

**void Pe0003_CbusDisable (LPC_SSP_T *  *pSSP*)**

Disable the C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |

**Returns:**

None

**void Pe0003_CbusEnable (LPC_SSP_T *  *pSSP*)**

Enable the C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |

**Returns:**

None

## void Pe0003_CbusGeneralReset (LPC_SSP_T * *pSSP*)

C-BUS general reset.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |

**Returns:**

None

## void Pe0003_CbusInit (LPC_SSP_T * *pSSP*)

Initialise the C-BUS port, using the default configuration.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1, CBUS2. |

**Returns:**

None

**Note:**

Legacy function - use Pe0003_CbusStart instead.

**Warning:**

Default frequency set at 10MHz SOME DEVICES WORK AT 5MHz

## void Pe0003_CbusReadBstream (LPC_SSP_T * *pSSP*, uint32_t *iAddr*, uint8_t * *pData*, uint32_t *iCount*)

Read a byte from a C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |
| *iAddr* | - C-Bus address |
| *pData* | - Data pointer to read data store |
| *iCount* | - Number of byte to read |

**Returns:**

None

uint32_t Pe0003_CbusReadByte (LPC_SSP_T * *pSSP*, uint32_t *iAddr*)

Read a byte from a C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |
| *iAddr* | - C-BUS address |

**Returns:**

Byte read

### uint32_t Pe0003_CbusReadWord (LPC_SSP_T * *pSSP*, uint32_t *iAddr*)

Read a word from a C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |
| *iAddr* | - C-BUS address |

**Returns:**

Word read

### void Pe0003_CbusReadWstream (LPC_SSP_T * *pSSP*, uint32_t *iAddr*, uint16_t * *pData*, uint32_t *iCount*)

Read a word from a C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |
| *iAddr* | - C-Bus address |
| *pData* | - Data pointer to read data |
| *iCount* | - Number of words to read |

**Returns:**

None

## void Pe0003_CbusRegisterRead (LPC_SSP_T * *pSSP*, uint8_t *cbus_address*, uint16_t * *data_ptr*, uint8_t *bytes_per_access*, uint8_t *accesses*)

Streaming read of bytes or words from a C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2 |
| *cbus_address* | - C-Bus register address to read |
| *data_ptr* | - Pointer to the buffer that stores the data |
| *bytes_per_access* | - 1 read bytes and 2 read words |
| *accesses* | - Number of data bytes or words to read |

**Returns:**

None

**Note:**

Legacy function use:
Pe0003_CbusReadByte
Pe0003_CbusReadWord
Pe0003_CbusReadBstream
Pe0003_CbusReadWstream

## void Pe0003_CbusRegisterWrite (LPC_SSP_T * *pSSP*, uint8_t *cbus_address*, uint16_t * *data_ptr*, uint8_t *bytes_per_access*, uint8_t *accesses*)

Streaming write of bytes or words to a C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1, CBUS2 |
| *cbus_address* | - C-BUS register address to write |
| *data_ptr* | - Pointer to the buffer that contains the data |
| *bytes_per_access* | - 1 write bytes and 2 write words |
| *accesses* | - Number of data bytes or words to write |

**Returns:**

None

**Note:**

Legacy function - use
Pe0003_CbusWriteNoData
Pe0003_CbusGeneralReset

Pe0003_CbusWriteByte
Pe0003_CbusWriteWord
Pe0003_CbusWriteBstream
Pe0003_CbusWriteWstream

### uint8_t Pe0003_CbusRxFifoEmpty (LPC_SSP_T * *pSSP*)

Check empty FIFO status.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2 |

**Returns:**

boolean - TRUE for empty

### void Pe0003_CbusSetFrequency (LPC_SSP_T * *pSSP*, uint32_t *frequency*)

Change the C-BUS frequency from the 10MHz default.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |
| *frequency* | - Value in Hz |

**Returns:**

None

### void Pe0003_CbusSetSpeed (LPC_SSP_T * *pSSP*, uint32_t *iSpeed*)

Set the speed of the C-BUS SCLK in Hz.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |
| *iSpeed* | - SCLK in Hz |

**Returns:**

None

### void Pe0003_CbusStart (LPC_SSP_T * *pSSP*)

Start the C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |

**Returns:**

None

**Warning:**

Default frequency set at 10MHz SOME DEVICES WORK AT 5MHz

### void Pe0003_CbusStop (LPC_SSP_T * *pSSP*)

Stop the C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |

**Returns:**

None

### void Pe0003_CbusWriteBstream (LPC_SSP_T * *pSSP*, uint32_t *iAddr*, uint8_t * *pData*, uint32_t *iCount*)

Write a byte stream to a C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |
| *iAddr* | - C-Bus address |
| *pData* | - Pointer to buffer that contains the data |
| *iCount* | - Number of bytes to write |

**Returns:**

None

### void Pe0003_CbusWriteByte (LPC_SSP_T * *pSSP*, uint32_t *iAddr*, uint32_t *iData*)

Write a byte to a C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |
| *iAddr* | - C-Bus address |
| *iData* | - Data bytes to write |

**Returns:**

None

### void Pe0003_CbusWriteNoData (LPC_SSP_T * *pSSP*, uint32_t *iAddr*)

C-BUS write no data to an specific address.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |
| *iAddr* | - Address to write |

**Returns:**

None

### void Pe0003_CbusWriteWord (LPC_SSP_T * *pSSP*, uint32_t *iAddr*, uint32_t *iData*)

Write a word to a C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |
| *iAddr* | - C-Bus address |
| *iData* | - Data word to write |

**Returns:**

None

### void Pe0003_CbusWriteWstream (LPC_SSP_T * *pSSP*, uint32_t *iAddr*, uint16_t * *pData*, uint32_t *iCount*)

Write a byte stream to a C-BUS port.

**Parameters:**

| | |
|---|---|
| *pSSP* | - CBUS1 or CBUS2. |
| *iAddr* | - C-BUS address |

| | |
|---|---|
| *pData* | - Pointer to buffer that contains the data |
| *iCount* | - Number of words to write |

**Returns:**

None

## 4.3 inc/ftdi.h File Reference

```
#include "lpc_types.h"
```

### 4.3.1 Macros

- #define **PE0003_USB_BAUD** 3000000
  *USB data rate.*

### 4.3.2 Functions

- void **Pe0003_UsbFtdiInit** (void)
  *Initialise the USB controller.*
- void **Pe0003_UsbFtdiWriteByte** (uint8_t data)
  *Write a byte into the USB controller.*
- void **Pe0003_UsbFtdiWriteArrayByte** (uint8_t *data, uint32_t len)
  *Write an array of bytes into the USB controller.*
- uint8_t **Pe0003_UsbFtdiReadByte** (void)
  *Read a byte from the USB controller.*
- void **Pe0003_UsbFtdiReadArrayByte** (uint8_t *data, uint32_t len)
  *Read an array of bytes into the USB controller.*
- uint8_t **Pe0003_UsbFtdiRxFifoEmpty** (void)
  *Read USB controller FIFO empty status.*
- uint8_t **Pe0003_UsbFtdiUartOverrun** (void)
  *Read USB controller UART overrun status.*
- void **Pe0003_UsbFtdiClearFifos** (void)
  *Clear USB FIFOs.*
- void **Pe0003_UsbSetInt** ()
  *Configure USB controller to use the interrupt system.*
- void **Pe0003_UsbIntEnable** ()
  *Enable USB interrupts.*
- void **Pe0003_UsbIntDisable** ()
  *Disable USB interrupts.*

### 4.3.3 Macro Definition Documentation

#### #define PE0003_USB_BAUD 3000000

USB data rate.

### 4.3.4 Function Documentation

#### void Pe0003_UsbFtdiClearFifos (void )

Clear USB FIFOs.

**Returns:**
None

### void Pe0003_UsbFtdiInit (void )

Initialise the USB controller.

**Returns:**
None

### void Pe0003_UsbFtdiReadArrayByte (uint8_t * *data*, uint32_t *len*)

Read an array of bytes into the USB controller.

**Parameters:**

| | |
|---|---|
| *data* | - array of bytes to read |
| *len* | - number of data to read |

**Returns:**
None

### uint8_t Pe0003_UsbFtdiReadByte (void )

Read a byte from the USB controller.

**Returns:**
Data byte read

### uint8_t Pe0003_UsbFtdiRxFifoEmpty (void )

Read USB controller FIFO empty status.

**Returns:**
Fifo empty TRUE

### uint8_t Pe0003_UsbFtdiUartOverrun (void )

Read USB controller UART overrun status.

**Returns:**
USB controller UART overrun status

### void Pe0003_UsbFtdiWriteArrayByte (uint8_t * *data*, uint32_t *len*)

Write an array of bytes into the USB controller.

**Parameters:**

| | |
|---|---|
| *data* | - array of bytes to write |
| *len* | - number of data to write |

**Returns:**

None

### void Pe0003_UsbFtdiWriteByte (uint8_t *data*)

Write a byte into the USB controller.

**Parameters:**

| | |
|---|---|
| *data* | - byte to write |

**Returns:**

None

### void Pe0003_UsbIntDisable ()

Disable USB interrupts.

**Returns:**

None

### void Pe0003_UsbIntEnable ()

Enable USB interrupts.

**Returns:**

None

### void Pe0003_UsbSetInt ()

Configure USB controller to use the interrupt system.

**Returns:**

None

## 4.4 inc/gpio.h File Reference

### 4.4.1 Data Structures

- struct **PE0003_IOBit**

### 4.4.2 *Used to remap the pins.* Macros

- #define **MAX_IO_BITS** 16
- #define **MAX_DIO_BITS** 4
- #define **MAX_GPIO_BITS** 8
- #define **MAX_DEDICATEDIO_BITS** 7

### 4.4.3 Enumerations

- enum **ioport** { **PE0003_IO** = 0, **PE0003_DIO**, **PE0003_GPIO** }
  *Description of the GPIO ports available in the PE0003*
- . enum **ded_iopin** { **BOOTEN11** = 0, **BOOTEN12**, **BOOTEN21**, **BOOTEN22**, **IRQN1**, **IRQN2**, **RS232CBUS** }

### 4.4.4 *Enumeration of the dedicated pins avaiable in the C-BUS.* Functions

- **__attribute** ((unused)) static **PE0003_IOBit** PE0003_IOMap[**MAX_IO_BITS**]
- void **Pe0003_GpioInit** ()
  *Initialise the GPIO port.*
- void **Pe0003_GpioSetDir** (uint16_t direction)
  *Set the pins direction of the GPIO port.*
- void **Pe0003_GpioWrite** (uint16_t data)
  *Write a value into the GPIO port.*
- uint32_t **Pe0003_GpioRead** ()
  *Read the GPIO port value.*
- void **Pe0003_GpioSetpin** (uint16_t pin)
  *Set a DIO pin high.*
- void **Pe0003_GpioClearpin** (uint16_t pin)
  *Set a DIO pin low.*
- void **Pe0003_DIOInit** ()
  *Initialise the DIO port.*
- void **Pe0003_DIOWrite** (uint16_t data)
  *Write to the DIO port pins.*
- void **Pe0003_DIOSetpin** (uint16_t pin)
  *Set a DIO port pin high.*
- void **Pe0003_DIOClearpin** (uint16_t pin)
  *Set a DIO port pin low.*
- void **Pe0003_IOInit** ()
  *Initialise the IO port.*
- void **Pe0003_IOSetDir** (uint16_t direction)
  *Set the direction of the IO port pins.*
- void **Pe0003_IOWrite** (uint16_t data)
  *Write a value to the IO port.*
- uint32_t **Pe0003_IORead** ()
  *Read the IO port.*
- void **Pe0003_IOSetpin** (uint16_t pin)
  *Set a DIO pin high.*
- void **Pe0003_IOClearpin** (uint16_t pin)
  *Set a DIO pin low.*

- void **Pe0003_DedicatedIOInit** ()
  *Initialise dedicated IO pins.*
- void **Pe0003_SetBooten1** (uint8_t cbus)
  *Set BOOTEN1 high.*
- void **Pe0003_SetBooten2** (uint8_t cbus)
  *Set BOOTEN2 high.*
- void **Pe0003_ClearBooten1** (uint8_t cbus)
  *Set BOOTEN1 low.*
- void **Pe0003_ClearBooten2** (uint8_t cbus)
  *Set BOOTEN2 low.*
- void **Pe0003_IrqnIntSet** ()
  *Set IRQN1 and IRQN2 interrupts (set pins high)*
- void **Pe0003_IrqnEnable** ()
  *Enable IRQN1 and IRQN2 interrupts.*
- void **Pe0003_IrqnDisable** ()
  *Disable IRQN1 and IRQN2 interrupts.*
- void **Pe0003_GpioDedicatedIOInit** ()
  *Initialise input/output dedicated C-BUS IO pins.*
- void **Pe0003_GpioDedicatedIODirSet** (enum **ded_iopin** iopin, uint32_t iodir)
  *Set the direction of C-BUS IO pin.*
- void **Pe0003_GpioDedicatedIOSet** (enum **ded_iopin** iopin)
  *Set C-BUS IO high.*
- void **Pe0003_GpioDedicatedIOClear** (enum **ded_iopin** iopin)
  *Set a C-BUS IO low.*
- uint8_t **Pe0003_GpioDedicatedIORead** (enum **ded_iopin** iopin)
  *Read dedicated input/output IO pin.*
- uint8_t **Pe0003_GpioDedicatedIOIntGetStatus** ()
  *Read both IRQN1 and IRQN2 interrupt pins at the same time (Created to use with the SH)*
- void **Pe0003_GpioDedicatedIOIntSet** ()
  *Set IRQN1 and IRQN2 interrupts (set pins high)*
- void **Pe0003_GpioDedicatedIOIntEnable** ()
  *Enable IRQN1 and IRQN2 interrupts.*
- void **Pe0003_GpioDedicatedIOIntDisable** ()
  *Disable IRQN1 IRQN2 interrupts.*
- void **Pe0003_GpioGenIOInit** ()
  *Initialise all the general IO ports.*
- void **Pe0003_GpioGenIOWritePin** (enum **ioport** pe0003IOPort, uint32_t pe0003IOPin, Bool setting)
  *Set the level of the ports pin.*
- uint32_t **Pe0003_GpioGenIOReadPin** (enum **ioport** pe0003IOPort, uint32_t pe0003IOPin)
  *Read the value of the specified port pin.*
- void **Pe0003_GpioGenIODirSetPin** (enum **ioport** pe0003IOPort, uint32_t pe0003IOPin, uint8_t setting)
  *Set the direction of the specified port pin.*
- void **Pe0003_GpioGenIOWrite** (enum **ioport** pe0003IOPort, uint32_t value)
  *Write a value directly to the specified port pin.*
- uint32_t **Pe0003_GpioGenIORead** (enum **ioport** pe0003IOPort)
  *Read a value from specified port.*
- void **Pe0003_GpioGenIOSetPin** (enum **ioport** pe0003IOPort, uint32_t pe0003IOPin)
  *Set the specified port pin to 1.*

- void **Pe0003_GpioGenIOClearPin** (enum **ioport** pe0003IOPort, uint32_t pe0003IOPin)
  *Set the specified port pin to 0.*
- void **Pe0003_GpioGenIOPortWriteDir** (enum **ioport** pe0003IOPort, uint16_t direction)
  *Set the direction of the output/input specified port pins.*
- void **ClearIRQN1Int** ()
  *Clear serviced interrupt inside the interrupt handler function for IRQN1.*
- void **ClearIRQN2Int** ()
  *Clear serviced interrupt inside the interrupt handler function for IRQN2.*

### 4.4.5 Macro Definition Documentation

**#define MAX_DEDICATEDIO_BITS 7**

**#define MAX_DIO_BITS 4**

**#define MAX_GPIO_BITS 8**

**#define MAX_IO_BITS 16**

PE0003 GPIO PORTS AND THE DEDICATED PINS

The ports are divided in two sections general io ports (GPIO, DIO, IO) and dedicated pins: (BOOTEN1_1, BOOTEN1_2, BOOTEN2_1, BOOTEN2_2, IRQN1, IRQN2, RS232CBUS)

General io ports and dedicated pins have their corresponding functions: General IO uses GPIO_IO_function_name, Dedicated IO uses GPIO_DI_function_name.

IO port - Pins shared with CBUS connectors
DIO port - Control the Leds D1,D2,D3,D4

Notice DIO1-4 GPIO port - General purpose IO pins (GPIO connector) Dedicated pins - Used to control configurations with the EV Kits and receive interrupts. All ports configured as inputs, except dedicated IO ports BOOTEN pins as output, IRQN pins as inputs.

THERE ARE TWO APIs

- API control, use of this last API is recommended
- Legacy API

### 4.4.6 Enumeration Type Documentation

#### enum ded_iopin

Enumeration of the dedicated pins avaiable in the C-BUS.

**Enumerator**

| | |
|---|---|
| ***BOOTEN11*** | BOOTEN1_1. |
| ***BOOTEN12*** | BOOTEN1_2. |
| ***BOOTEN21*** | BOOTEN2_1. |
| ***BOOTEN22*** | BOOTEN2_2. |
| ***IRQN1*** | IRQN1. |
| ***IRQN2*** | IRQN2. |
| ***RS232CBUS*** | RS232CBUS. |

#### enum ioport

Description of the GPIO ports available in the PE0003

.

**Enumerator**

| | |
|---|---|
| ***PE0003_IO*** | PE0003_IO ///<GPIOs available from the C-BUS ports. |
| ***PE0003_DIO*** | PE0003_DIO ///<GPIOs connected to the LEDs on the board. |
| ***PE0003_GPIO*** | PE0003_GPIO ///<GPIOs available from the GPIO port. |

### 4.4.7 Function Documentation

#### void ClearIRQN1Int ()

Clear serviced interrupt inside the interrupt handler function for IRQN1.

#### void ClearIRQN2Int ()

Clear serviced interrupt inside the interrupt handler function for IRQN2.

#### void Pe0003_ClearBooten1 (uint8_t *cbus*)

Set BOOTEN1 low.

**Parameters:**

| | |
|---|---|
| *cbus* | - 1 for C-BUS1 and 2 for C-BUS2 |

**Returns:**

None

### void Pe0003_ClearBooten2 (uint8_t *cbus*)

Set BOOTEN2 low.

**Parameters:**

| | |
|---|---|
| *cbus* | - 1 for C-BUS1 and 2 for C-BUS2 |

**Returns:**

None

### void Pe0003_DedicatedIOInit ()

Initialise dedicated IO pins.

**Returns:**

None

**Note:**

Configures dedicated pins such as BOOTEN1/2 and IRQN1/2 on the input/output ports

### void Pe0003_DIOClearpin (uint16_t *pin*)

Set a DIO port pin low.

**Parameters:**

| | |
|---|---|
| *pin* | - pin to clear. Values 0-3 |

**Returns:**

None

### void Pe0003_DIOInit ()

Initialise the DIO port.

**Returns:**

None

### void Pe0003_DIOSetpin (uint16_t *pin*)

Set a DIO port pin high.

**Parameters:**

| | |
|---|---|
| *pin* | - pin to set high: Values 0-3 |

**Returns:**

None

### void Pe0003_DIOWrite (uint16_t *data*)

Write to the DIO port pins.

**Parameters:**

| | |
|---|---|
| *data* | - Value to write |

**Returns:**

None

### void Pe0003_GpioClearpin (uint16_t *pin*)

Set a DIO pin low.

**Parameters:**

| | |
|---|---|
| *pin* | - pin to clear. 0-7 |

**Returns:**

None

### void Pe0003_GpioDedicatedIOClear (enum ded_iopin *iopin*)

Set a C-BUS IO low.

**Parameters:**

| | |
|---|---|
| *iopin* | - Name of the dedicated pin BOOTEN11, BOOTEN12, BOOTEN21, BOOTEN22, IRQN1, IRQN2 |

**Returns:**

None

### void Pe0003_GpioDedicatedIODirSet (enum ded_iopin *iopin*, uint32_t *iodir*)

Set the direction of C-BUS IO pin.

**Parameters:**

| | |
|---|---|
| *iopin* | - Name of the dedicated pin BOOTEN11, BOOTEN12, BOOTEN21, BOOTEN22, IRQN1, IRQN2 |
| *iodir* | - 1 for output, 0 for intput |

**Returns:**
None

### void Pe0003_GpioDedicatedIOInit ()

Initialise input/output dedicated C-BUS IO pins.

**Returns:**
None

### void Pe0003_GpioDedicatedIOIntDisable ()

Disable IRQN1 IRQN2 interrupts.

**Returns:**
None

### void Pe0003_GpioDedicatedIOIntEnable ()

Enable IRQN1 and IRQN2 interrupts.

**Returns:**
None

### uint8_t Pe0003_GpioDedicatedIOIntGetStatus ()

Read both IRQN1 and IRQN2 interrupt pins at the same time (Created to use with the SH)

**Returns:**
Returns two bits - bit 0 for IRQN1(mask 0x01) and bit 1 for IRQN2(mask 0x02)

**Note:**
This function uses polling method to fetch the value of the IRQN1/2 pins. Ideally, an interrupt handler triggered by real interrupt is recommended. See functions:
Pe0003_GpioDedicatedIOIntEnable - Enable real interrupts
Pe0003_GpioDedicatedIOIntSet - Set a real interrupt.
Pe0003_GpioDedicatedIOIntDisable - Disable real interrupts
Handlers to use
GPIO0_IRQHandler - Handler for IRQN1
GPIO1_IRQHandler - Handler for IRQN2

```
    //interrupt handler for IRQN1
void GPIO0_IRQHandler(void) {

    //DO SOME STUFF


        ClearIRQN1Int();

}

```

```
//interrupt handler for IRQN2
void GPIO1_IRQHandler(void) {

    //DO SOME STUFF

    ClearIRQN2Int();

}
```

### void Pe0003_GpioDedicatedIOIntSet ()

Set IRQN1 and IRQN2 interrupts (set pins high)

**Returns:**

None

**Note:**

This function uses polling method to fetch the value of the IRQN1/2 pins. Ideally, an interrupt handler triggered by real interrupt is recommended. See functions:
Pe0003_GpioDedicatedIOIntEnable - Enable real interrupts
Pe0003_GpioDedicatedIOIntSet - Set a real interrupt.
Pe0003_GpioDedicatedIOIntDisable - Disable real interrupts
Handlers to use
GPIO0_IRQHandler - Handler for IRQN1
GPIO1_IRQHandler - Handler for IRQN2

```
    //interrupt handler for IRQN1
void GPIO0_IRQHandler(void) {

    //DO SOME STUFF


        ClearIRQN1Int();

}

//interrupt handler for IRQN2
void GPIO1_IRQHandler(void) {

    //DO SOME STUFF

    ClearIRQN2Int();

}
```

### uint8_t Pe0003_GpioDedicatedIORead (enum ded_iopin *iopin*)

Read dedicated input/output IO pin.

**Parameters:**

| | |
|---|---|
| *iopin* | - Name of the dedicated pin BOOTEN11, BOOTEN12, BOOTEN21, BOOTEN22, IRQN1, IRQN2 |

**Returns:**

value of the pin

### void Pe0003_GpioDedicatedIOSet (enum ded_iopin *iopin*)

Set C-BUS IO high.

**Parameters:**

| | |
|---|---|
| *iopin* | - Name of the dedicated pin BOOTEN11, BOOTEN12, BOOTEN21, BOOTEN22, IRQN1, IRQN2 |

**Returns:**

None

### void Pe0003_GpioGenIOClearPin (enum ioport *pe0003IOPort*, uint32_t *pe0003IOPin*)

Set the specified port pin to 0.

**Parameters:**

| | |
|---|---|
| *pe0003IOPort* | - Port to use. Values PE0003_IO, PE0003_GPIO or PE0003_DIO |
| *pe0003IOPin* | - Pin to clear. 0 to 7 for GPIOs, 0 to 15 for IOs and 0 to 4 for DIOs |

**Returns:**

None

### void Pe0003_GpioGenIODirSetPin (enum ioport *pe0003IOPort*, uint32_t *pe0003IOPin*, uint8_t *setting*)

Set the direction of the specified port pin.

**Parameters:**

| | |
|---|---|
| *pe0003IOPort* | - Port to use. PE0003_IO, PE0003_GPIO or PE0003_DIO |
| *pe0003IOPin* | - Pin to read. Values from 0 to 7 for GPIOs, from 0 to 15 for IOs and from 0 to 4 for DIOs |
| *setting* | - 0 for input, 1 for output |

**Returns:**

None

### void Pe0003_GpioGenIOInit ()

Initialise all the general IO ports.

**Returns:**

None

**Note:**

The GPIO and IO port are configured as outputs. The DIO port is configured as inputs.

## void Pe0003_GpioGenIOPortWriteDir (enum ioport *pe0003IOPort*, uint16_t *direction*)

Set the direction of the output/input specified port pins.

**Parameters:**

| | |
|---|---|
| *pe0003IOPort* | - Port to use. PE0003_IO, PE0003_GPIO or PE0003_DIO |
| *direction* | - 1 for output and 0 for input. |

**Returns:**

Nothing

```
Pe0003_GpioGenIOPortWriteDir(PE0003_GPIO, 0x04);
Sets the GPIO0-3 as outputs and GPIO4-7 as inputs
```

## uint32_t Pe0003_GpioGenIORead (enum ioport *pe0003IOPort*)

Read a value from specified port.

**Parameters:**

| | |
|---|---|
| *pe0003IOPort* | - Port to use. PE0003_IO, PE0003_GPIO or PE0003_DIO |

**Returns:**

value from the target port

## uint32_t Pe0003_GpioGenIOReadPin (enum ioport *pe0003IOPort*, uint32_t *pe0003IOPin*)

Read the value of the specified port pin.

**Parameters:**

| | |
|---|---|
| *pe0003IOPort* | - Port to use. PE0003_IO, PE0003_GPIO or PE0003_DIO |
| *pe0003IOPin* | - Pin to read. 0 to 7 for GPIOs, 0 to 15 for IOs and 0 to 4 for DIOs |

**Returns:**

Pin value

## void Pe0003_GpioGenIOSetPin (enum ioport *pe0003IOPort*, uint32_t *pe0003IOPin*)

Set the specified port pin to 1.

**Parameters:**

| | |
|---|---|
| *pe0003IOPort* | - Port to use. PE0003_IO, PE0003_GPIO or PE0003_DIO |
| *pe0003IOPin* | - Pin to read. 0 to 7 for GPIOs, 0 to 15 for IOs and 0 to 4 for DIOs |

**Returns:**

None

### void Pe0003_GpioGenIOWrite (enum ioport *pe0003IOPort*, uint32_t *value*)

Write a value directly to the specified port pin.

**Parameters:**

| | |
|---|---|
| *pe0003IOPort* | - Port to use. PE0003_IO, PE0003_GPIO or PE0003_DIO |
| *value* | - Value to write |

**Returns:**

None

### void Pe0003_GpioGenIOWritePin (enum ioport *pe0003IOPort*, uint32_t *pe0003IOPin*, Bool *setting*)

Set the level of the ports pin.

**Parameters:**

| | |
|---|---|
| *pe0003IOPort* | - Port to use. PE0003_IO, PE0003_GPIO or PE0003_DIO) |
| *pe0003IOPin* | - Pin to set or clear. 0 to 7 for GPIOs, 0 to 15 for IOs and 0 to 4 for DIOs |
| *setting* | - TRUE or FALSE to set the pin respectively |

**Returns:**

None

### void Pe0003_GpioInit ()

Initialise the GPIO port.

**Returns:**

None

### uint32_t Pe0003_GpioRead ()

Read the GPIO port value.

**Returns:**

GPIO value

### void Pe0003_GpioSetDir (uint16_t *direction*)

Set the pins direction of the GPIO port.

**Parameters:**

| | |
|---|---|
| *direction* | - 1 for output 0 for input |

**Returns:**

None

```
//Sets the GPIO0-3 as inputs and GPIO4-7 as outputs
Pe0003_GpioSetDir(0x40);
```

### void Pe0003_GpioSetpin (uint16_t *pin*)

Set a DIO pin high.

**Parameters:**

| | |
|---|---|
| *pin* | - pin to set high. 0-7 |

**Returns:**

None

### void Pe0003_GpioWrite (uint16_t *data*)

Write a value into the GPIO port.

**Parameters:**

| | |
|---|---|
| *data* | - value to write |

**Returns:**

None

### void Pe0003_IOClearpin (uint16_t *pin*)

Set a DIO pin low.

**Parameters:**

| | |
|---|---|
| *pin* | - pin to clear. 0-15 |

**Returns:**

None

### void Pe0003_IOInit ()

Initialise the IO port.

**Returns:**

None

### uint32_t Pe0003_IORead ()

Read the IO port.

**Returns:**

Value read from the IO port

### void Pe0003_IOSetDir (uint16_t *direction*)

Set the direction of the IO port pins.

**Parameters:**

| | |
|---|---|
| *direction* | - 1 for output, 0 for input |

**Returns:**

None

### void Pe0003_IOSetpin (uint16_t *pin*)

Set a DIO pin high.

**Parameters:**

| | |
|---|---|
| *pin* | - pin to set high. 0-15 |

**Returns:**

None

### void Pe0003_IOWrite (uint16_t *data*)

Write a value to the IO port.

**Parameters:**

| | |
|---|---|
| *data* | - value to write |

**Returns:**

None

## void Pe0003_IrqnDisable ()

Disable IRQN1 and IRQN2 interrupts.

**Returns:**

None

## void Pe0003_IrqnEnable ()

Enable IRQN1 and IRQN2 interrupts.

**Returns:**

None

## void Pe0003_IrqnIntSet ()

Set IRQN1 and IRQN2 interrupts (set pins high)

**Returns:**

None

**Note:**

This function uses polling method to fetch the value of the IRQN1/2 pins. Ideally, an interrupt handler triggered by real interrupt is recommended. See functions: Pe0003_GpioDedicatedIOIntEnable - Enable real interrupts Pe0003_GpioDedicatedIOIntSet - Set a real interrupt. Pe0003_GpioDedicatedIOIntDisable - Disable real interrupts Handlers to use GPIO0_IRQHandler - Handler for IRQN1 GPIO1_IRQHandler - Handler for IRQN2

```
    //interrupt handler for IRQN1
void GPIO0_IRQHandler(void) {

    //DO SOME STUFF


        ClearIRQN1Int();

}

//interrupt handler for IRQN2
void GPIO1_IRQHandler(void) {

    //DO SOME STUFF

    ClearIRQN2Int();

}
```

### void Pe0003_SetBooten1 (uint8_t *cbus*)

Set BOOTEN1 high.

**Parameters:**

| | |
|---|---|
| *cbus* | - 1 for C-BUS1 and 2 for C-BUS2 |

**Returns:**

None

### void Pe0003_SetBooten2 (uint8_t *cbus*)

Set BOOTEN2 high.

**Parameters:**

| | |
|---|---|
| *cbus* | - 1 for C-BUS1 and 2 for C-BUS2 |

**Returns:**

None

## 4.5 inc/hostport_i2s.h File Reference

`#include "chip.h"`

### 4.5.1 Macros

- #define **CBUS_SPEED** 11000000
- #define **I2S_TX_BITRATE_DFT** 1000000
- #define **I2S_RX_BITRATE_DFT** 1000000
- #define **PE_I2S_WORDWIDTH_8** 0
- #define **PE_I2S_WORDWIDTH_16** 1
- #define **PE_I2S_WORDWIDTH_32** 3
- #define **PE_I2S_STEREO** 0
- #define **PE_I2S_MONO** 1
- #define **PE_I2S_MASTER_MODE** 0
- #define **PE_I2S_SLAVE_MODE** 1
- #define **PE_I2S_DMA_MODE** 0
- #define **PE_I2S_NODMA_MODE** 1
- #define **PE_I2S_TX** 0
- #define **PE_I2S_RX** 1
- #define **NORMAL_I2S**
- #define **NORMAL_I2S_BUFFER_SIZE** 20
- #define **I2S_BUFFER_SIZE NORMAL_I2S_BUFFER_SIZE**

### 4.5.2 Functions

- void **Pe0003_HostPortI2sGenInit** ()
  *Initialise the I2S port for normal operation as master.*
- void **Pe0003_HostPortI2sGenSlaveInit** ()
  *Initialise the I2S port for normal operation as slave.*
- void **Pe0003_HostPortI2sDeInit** ()
  *Deinitialise the I2S controller.*
- void **Pe0003_HostPortI2sStop** ()
  *Stop the I2S controller.*
- void **Pe0003_HostPortI2sStart** ()
  *Start the I2S controller.*

### 4.5.3 Variables

- uint32_t **I2sTx_Buf** [**I2S_BUFFER_SIZE**]
- uint32_t **I2sRx_Buf** [**I2S_BUFFER_SIZE**]
- uint8_t **isDmaTxCompleted**
- uint8_t **isDmaRxCompleted**
- uint32_t **isWrongRx**
- uint32_t **isWrongTx**

### 4.5.4 Macro Definition Documentation

#### #define CBUS_SPEED 11000000

#### #define I2S_BUFFER_SIZE NORMAL_I2S_BUFFER_SIZE

#### #define I2S_RX_BITRATE_DFT 1000000

**#define I2S_TX_BITRATE_DFT 1000000**

**#define NORMAL_I2S**

Three modes of operation
NORMAL_I2S
DMA_I2S
IRQ_I2S
NORMAL_I2S by default

**#define NORMAL_I2S_BUFFER_SIZE 20**

**#define PE_I2S_DMA_MODE 0**

**#define PE_I2S_MASTER_MODE 0**

**#define PE_I2S_MONO 1**

**#define PE_I2S_NODMA_MODE 1**

**#define PE_I2S_RX 1**

**#define PE_I2S_SLAVE_MODE 1**

**#define PE_I2S_STEREO 0**

**#define PE_I2S_TX 0**

**#define PE_I2S_WORDWIDTH_16 1**

**#define PE_I2S_WORDWIDTH_32 3**

**#define PE_I2S_WORDWIDTH_8 0**

### 4.5.5 Function Documentation

**void Pe0003_HostPortI2sDeInit ()**

Deinitialise the I2S controller.

**Returns:**

None

### void Pe0003_HostPortI2sGenInit ()

Initialise the I2S port for normal operation as master.

NORMAL OPERATION

**Returns:**

None

**Note:**

Initialise the host port in Stereo, Wordwidth32 and master frequency set at 1MHz

### void Pe0003_HostPortI2sGenSlaveInit ()

Initialise the I2S port for normal operation as slave.

**Returns:**

None

**Note:**

Initialise the host port in Stereo, Wordwidth16 and slave

### void Pe0003_HostPortI2sStart ()

Start the I2S controller.

**Returns:**

None

#### void Pe0003_HostPortI2sStop ()

Stop the I2S controller.

**Returns:**
None

### 4.5.6 Variable Documentation

#### uint32_t I2sRx_Buf[I2S_BUFFER_SIZE]

#### uint32_t I2sTx_Buf[I2S_BUFFER_SIZE]

#### uint8_t isDmaRxCompleted

#### uint8_t isDmaTxCompleted

#### uint32_t isWrongRx

#### uint32_t isWrongTx

## 4.6 inc/hostport_uart.h File Reference

```
#include "lpc_types.h"
```

### 4.6.1 Macros

- #define **HOSTPORTUART_BAUD** 115200

### 4.6.2 Functions

- void **Pe0003_HostPortUartInit** (void)
  *Initialise USART2 controller and GPIO on the HostPort.*
- void **Pe0003_HostPortUartWriteByte** (uint8_t byte)
  *Write a byte to the USART2 controller.*
- uint8_t **Pe0003_HostPortUartReadByte** (void)
  *Read a byte from the USART controller.*
- uint8_t **Pe0003_HostPortUartLsr** (void)
  *Get the line status of the USART controller.*
- uint8_t **Pe0003_HostPortUartRxFifoEmpty** (void)
  *Check if USART controller's Rx FIFO is empty.*
- void **Pe0003_HostPortUartClearFifos** (void)
  *Clear the USART fifos.*
- void **Pe0003_HostPortGpioInit** ()
  *Initialize the Gpios as inputs.*
- void **Pe0003_HostPortGpioSetDirRTSN** (uint8_t direction)
  *Set the RTSN pin direction.*
- void **Pe0003_HostPortGpioSetDirPTTN** (uint8_t direction)
  *Set the PTTN pin direction.*
- void **Pe0003_HostPortGpioWriteRTSN** (uint8_t data)
  *Set the RTSN pin high or low.*
- void **Pe0003_HostPortGpioWritePTTN** (uint8_t data)
  *Set the PTTN pin high or low.*
- uint8_t **Pe0003_HostPortGpioReadRTSN** ()
  *Read RTSN Gpio pin.*
- uint8_t **Pe0003_HostPortGpioReadPTTN** ()
  *Read PTTN Gpio pin.*

### 4.6.3 Macro Definition Documentation

#### #define HOSTPORTUART_BAUD 115200

To use the HOSTPORT_UART some pins need to be reconfigured in **pe0003.h**

### 4.6.4 Function Documentation

#### void Pe0003_HostPortGpioInit ()

Initialize the Gpios as inputs.

**Returns:**

None

**Note:**

The pin multiplexor in **PE0003.h** library must be configured correctly before using this function Check configuration

### uint8_t Pe0003_HostPortGpioReadPTTN ()

Read PTTN Gpio pin.

**Returns:**

None

### uint8_t Pe0003_HostPortGpioReadRTSN ()

Read RTSN Gpio pin.

**Returns:**

None

### void Pe0003_HostPortGpioSetDirPTTN (uint8_t *direction*)

Set the PTTN pin direction.

**Parameters:**

| | |
|---|---|
| *direction* | - 1 for output, 0 for input |

**Returns:**

None

### void Pe0003_HostPortGpioSetDirRTSN (uint8_t *direction*)

Set the RTSN pin direction.

**Parameters:**

| | |
|---|---|
| *direction* | - 1 for output, 0 for input |

**Returns:**

None

### void Pe0003_HostPortGpioWritePTTN (uint8_t *data*)

Set the PTTN pin high or low.

**Parameters:**

| | |
|---|---|
| *data* | - 1 set high, 0 set low |

**Returns:**

None

### void Pe0003_HostPortGpioWriteRTSN (uint8_t *data*)

Set the RTSN pin high or low.

**Parameters:**

| | |
|---|---|
| *data* | - 1 set high, 0 set low |

**Returns:**

None

### void Pe0003_HostPortUartClearFifos (void )

Clear the USART fifos.

**Returns:**

None

### void Pe0003_HostPortUartInit (void )

Initialise USART2 controller and GPIO on the HostPort.

**Returns:**

None

**Note:**

To make use of this Hostport configuration, set the macro HOSTPORT_U2S_ENABLE Configuration

- No parity, 8bits, 1 stop bit
- HOSTPORTUART_BAUD
- RTS, CTS flow control
- FIFOs enabled

### uint8_t Pe0003_HostPortUartLsr (void )

Get the line status of the USART controller.

**Returns:**

USART controller status

### uint8_t Pe0003_HostPortUartReadByte (void )

Read a byte from the USART controller.

**Returns:**

Byte read

### uint8_t Pe0003_HostPortUartRxFifoEmpty (void )

Check if USART controller's Rx FIFO is empty.

**Returns:**

Return 1 if no data, otherwise 0

### void Pe0003_HostPortUartWriteByte (uint8_t *byte*)

Write a byte to the USART2 controller.

**Parameters:**

| | |
|---|---|
| *byte* | - byte to write |

**Returns:**

None

### 4.7 inc/pe0003.h File Reference

```
#include "lpc_types.h"
#include "chip.h"
#include <stdio.h>
```

#### 4.7.1 Macros

- #define **OK** 1
- #define **FAIL** 0
- #define **delay_us**(x) Timer_DelayUs(x)
- #define **EMC_IO** (SCU_MODE_REPEATER | SCU_MODE_HIGHSPEEDSLEW_EN | SCU_MODE_INBUFF_EN | SCU_MODE_ZIF_DIS)
- #define **LCD_PINCONFIG** (SCU_MODE_INACT | SCU_MODE_HIGHSPEEDSLEW_EN | SCU_MODE_INBUFF_EN | SCU_MODE_ZIF_DIS)
- #define **CLK_IN** (SCU_MODE_REPEATER | SCU_MODE_HIGHSPEEDSLEW_EN | SCU_MODE_INBUFF_EN | SCU_MODE_ZIF_DIS)
- #define **CLK_OUT** (SCU_MODE_REPEATER | SCU_MODE_HIGHSPEEDSLEW_EN | SCU_MODE_INBUFF_EN | SCU_MODE_ZIF_DIS)
- #define **GPIO_PUP** (SCU_MODE_PULLUP | SCU_MODE_INBUFF_EN )
- #define **GPIO_PDN** (SCU_MODE_PULLDOWN | SCU_MODE_INBUFF_EN )
- #define **GPIO_NOPULL** (SCU_MODE_INACT | SCU_MODE_INBUFF_EN )
- #define **UART_RX_TX** (SCU_MODE_REPEATER | SCU_MODE_INBUFF_EN )
- #define **SSP_IO** (SCU_MODE_REPEATER | SCU_MODE_HIGHSPEEDSLEW_EN | SCU_MODE_INBUFF_EN | SCU_MODE_ZIF_DIS)
- #define **LED_D1_SCU_PORT** 5
- #define **LED_D1_SCU_PIN** 0
- #define **LED_D1_PORT** 2
- #define **LED_D1_PIN** 9
- #define **LED_D1** 0x10
- #define **LED_D2_SCU_PORT** 5
- #define **LED_D2_SCU_PIN** 1
- #define **LED_D2_PORT** 2
- #define **LED_D2_PIN** 10
- #define **LED_D2** 0x11
- #define **LED_D3_SCU_PORT** 5
- #define **LED_D3_SCU_PIN** 3
- #define **LED_D3_PORT** 2
- #define **LED_D3_PIN** 12
- #define **LED_D3** 0x12
- #define **LED_D4_SCU_PORT** 5
- #define **LED_D4_SCU_PIN** 5
- #define **LED_D4_PORT** 2
- #define **LED_D4_PIN** 14
- #define **LED_D4** 0x13
- #define **U0_TXD_SCU_PORT** 2
- #define **U0_TXD_SCU_PIN** 0
- #define **U0_RXD_SCU_PORT** 2
- #define **U0_RXD_SCU_PIN** 1
- #define **U0_RTS_SCU_PORT** 5
- #define **U0_RTS_SCU_PIN** 2
- #define **U0_RTS_PORT** 2
- #define **U0_RTS_PIN** 11
- #define **U0_CTS_SCU_PORT** 5
- #define **U0_CTS_SCU_PIN** 4
- #define **U0_CTS_PORT** 2
- #define **U0_CTS_PIN** 13

- #define **U1_TXD_SCU_PORT** 1
- #define **U1_TXD_SCU_PIN** 13
- #define **U1_RXD_SCU_PORT** 1
- #define **U1_RXD_SCU_PIN** 14
- #define **U1_RTS_SCU_PORT** 5
- #define **U1_RTS_SCU_PIN** 2
- #define **U1_CTS_SCU_PORT** 5
- #define **U1_CTS_SCU_PIN** 4
- #define **U2_TXD_SCU_PORT** 2
- #define **U2_TXD_SCU_PIN** 10
- #define **U2_RXD_SCU_PORT** 2
- #define **U2_RXD_SCU_PIN** 11
- #define **SSP0_SCK_SCU_PORT** 3
- #define **SSP0_SCK_SCU_PIN** 0
- #define **SSP0_SSEL_SCU_PORT** 3
- #define **SSP0_SSEL_SCU_PIN** 6
- #define **SSP0_MOSI_SCU_PORT** 3
- #define **SSP0_MOSI_SCU_PIN** 8
- #define **SSP0_MISO_SCU_PORT** 3
- #define **SSP0_MISO_SCU_PIN** 7
- #define **SET_SSP0** 0x40
- #define **SSP0_SSEL_GPIO_PORT** 0
- #define **SSP0_SSEL_GPIO_PIN** 6
- #define **SSP0_SSEL_GPIO_MASK** (1 << **SSP0_SSEL_GPIO_PIN**)
- #define **SSP1_SCK_SCU_PORT** CLK0
- #define **SSP1_SCK_SCU_PIN** CLK0
- #define **SSP1_SSEL_SCU_PORT** 1
- #define **SSP1_SSEL_SCU_PIN** 5
- #define **SSP1_MOSI_SCU_PORT** 1
- #define **SSP1_MOSI_SCU_PIN** 4
- #define **SSP1_MISO_SCU_PORT** 1
- #define **SSP1_MISO_SCU_PIN** 3
- #define **SET_SSP1** 0x41
- #define **SSP1_SSEL_GPIO_PORT** 1
- #define **SSP1_SSEL_GPIO_PIN** 8
- #define **SSP1_SSEL_GPIO_MASK** (1 << **SSP1_SSEL_GPIO_PIN**)
- #define **S_ND** 0xFF
- #define **M_GPIO GPIO_NOPULL**
- #define **SCU_IO0** 1,0,**M_GPIO**,FUNC0
- #define **SCU_IO1** 6,2,**M_GPIO**,FUNC0
- #define **SCU_IO2** 6,3,**M_GPIO**,FUNC0
- #define **SCU_IO3** 6,5,**M_GPIO**,FUNC0
- #define **SCU_IO4** 7,4,**M_GPIO**,FUNC0
- #define **SCU_IO5** 7,5,**M_GPIO**,FUNC0
- #define **SCU_IO6** 7,6,**M_GPIO**,FUNC0
- #define **SCU_IO7** 7,0,**M_GPIO**,FUNC0
- #define **SCU_IO8** 7,3,**M_GPIO**,FUNC0
- #define **SCU_IO9** 6,9,**M_GPIO**,FUNC0
- #define **SCU_IO10** 6,10,**M_GPIO**,FUNC0
- #define **SCU_IO11** 6,11,**M_GPIO**,FUNC0
- #define **SCU_IO12** 5,7,**M_GPIO**,FUNC0
- #define **SCU_IO13** 2,2,**M_GPIO**,FUNC4
- #define **SCU_IO14** 2,3,**M_GPIO**,FUNC4
- #define **SCU_IO15** 2,5,**M_GPIO**,FUNC4
- #define **SCU_GPIO0** 4,0,**M_GPIO**,FUNC0
- #define **SCU_GPIO1** 4,1,**M_GPIO**,FUNC0

- #define **SCU_GPIO2** 4,2,**M_GPIO**,FUNC0
- #define **SCU_GPIO3** 4,3,**M_GPIO**,FUNC0
- #define **SCU_GPIO4** 4,4,**M_GPIO**,FUNC0
- #define **SCU_GPIO5** 4,5,**M_GPIO**,FUNC0
- #define **SCU_GPIO6** 4,6,**M_GPIO**,FUNC0
- #define **SCU_GPIO7** 3,5,**M_GPIO**,FUNC0
- #define **SCU_DIO1** 5,0,**M_GPIO**,FUNC0
- #define **SCU_DIO2** 5,1,**M_GPIO**,FUNC0
- #define **SCU_DIO3** 5,3,**M_GPIO**,FUNC0
- #define **SCU_DIO4** 5,5,**M_GPIO**,FUNC0
- #define **SCU_BOOTEN1_1** 4,9,**M_GPIO**,FUNC4
- #define **SCU_BOOTEN1_2** 4,10,**M_GPIO**,FUNC4
- #define **SCU_BOOTEN2_1** 6,7,**M_GPIO**,FUNC4
- #define **SCU_BOOTEN2_2** 6,8,**M_GPIO**,FUNC4
- #define **SCU_IRQN1** 4,8,**GPIO_PUP**,FUNC4
- #define **SCU_IRQN2** 5,6,**GPIO_PUP**,FUNC0
- #define **SD_CLK_SCU_PORT** CLK2
- #define **SD_CLK_SCU_PIN** CLK2
- #define **SD_CMD_SCU_PORT** 1
- #define **SD_CMD_SCU_PIN** 6
- #define **SD_DAT0_SCU_PORT** 1
- #define **SD_DAT0_SCU_PIN** 9
- #define **SD_DAT1_SCU_PORT** 1
- #define **SD_DAT1_SCU_PIN** 10
- #define **SD_DAT2_SCU_PORT** 1
- #define **SD_DAT2_SCU_PIN** 11
- #define **SD_DAT3_SCU_PORT** 1
- #define **SD_DAT3_SCU_PIN** 12
- #define **M_ETH** MD_PLN_FAST
- #define **SCU_ENET_RX_D** 1,16,**M_ETH**,FUNC7
- #define **SCU_ENET_MDC** 7,7,**M_ETH**,FUNC6
- #define **SCU_ENET_MDIO** 1,17,**M_ETH**,FUNC3
- #define **SCU_ENET_RXD0** 1,15,**M_ETH**,FUNC3
- #define **SCU_ENET_RXD1** 0,0,**M_ETH**,FUNC2
- #define **SCU_ENET_REF_CLK** 1,19,**M_ETH**,FUNC0
- #define **SCU_ENET_TXD0** 1,18,**M_ETH**,FUNC3
- #define **SCU_ENET_TXD1** 1,20,**M_ETH**,FUNC3
- #define **SCU_ENET_TX_EN** 0,1,**M_ETH**,FUNC6
- #define **M_I2S** SCU_PINIO_FAST
- #define **I2S_RXCLK** 0xF,4,M_I2S,FUNC7
- #define **I2S_RXD** 3,2,**M_I2S**,FUNC1
- #define **I2S_RX_WS** 3,1,**M_I2S**,FUNC1
- #define **I2S_TXCLK** 4,7,**M_I2S**,FUNC7
- #define **I2S_TXD** 7,2,**M_I2S**,FUNC2
- #define **I2S_WS** 7,1,**M_I2S**,FUNC2
- #define **HP_U2_TXD** 7,1,**UART_RX_TX**,FUNC6
- #define **HP_U2_RXD** 7,2,**UART_RX_TX**,FUNC6
- #define **HP_RTSN** 3,2,**GPIO_NOPULL**,FUNC4
- #define **HP_PTTN** 3,1,**GPIO_NOPULL**,FUNC4

### 4.7.2 Functions

- void **Pe0003_BoardInit** ()
  *Configure all the peripheral pins of the PE0003 board and set the clock.*
- void **Pe0003_BoardLedInit** ()
  *Initialise the LEDs.*

- void **Pe0003_BoardLedSet** (uint8_t ledNumber, uint8_t state)
  *Control a LED.*
- void **Pe0003_BoardLedToggle** (uint8_t ledNumber)
  *Toggle a LED state.*
- void **Pe0003_BoardUsartInit** (LPC_USART_T *pUART)
  *Initialise the pins for an UART/USART.*
- void **Pe0003_BoardUsartDeInit** (LPC_USART_T *pUART)
  *Deinitialise UART/USART.*
- void **Pe0003_BoardSdCardInit** ()
  *Initialise the pins for the SD Card controller.*
- void **Pe0003_BoardSspInit** (LPC_SSP_T *pSSP)
  *Initialise the SSP port pins, used for CBUS.*
- void **Pe0003_BoardI2sInit** ()
  *Initialise the HostPort I2S pins.*
- void **Pe0003_BoardGpioInit** ()
  *Initialise the GPIO pins used by the controllers.*
- void **Pe0003_BoardEthernetSet** ()
  *Initialise the Ethernet port pins.*
- void **Pe0003_InitClock** ()
  *Clock Initialisation. Set the main frequency to the maximum of 204MHz.*
- void **Pe0003_BoardHostPortUart2Init** ()
  *Initialise the Host port a UART configuration.*
- void **Pe0003_DbgInit** ()
  *Configure UART2 to use COM port for debugging. Note that: It is possible to use the NXP debugging framework from NXP with the proper if correctly configured. Other options for debugging include using the semihost configuration with printf, scanf functions and the LPCXpresso console.*
- void **Pe0003_DbgUartPutStr** (const void *str)
  *Debug port function put a string.*
- void **Pe0003_DbgUartPutChar** (uint8_t val)
  *Debug port function put char.*
- void **Pe0003_DbgUartGetStr** (void *str)
  *Debug port funtion. Get a string.*
- uint8_t **Pe0003_DbgUartGetChar** ()
  *Debug port function. Get char.*

### 4.7.3 Macro Definition Documentation

**#define CLK_IN (SCU_MODE_REPEATER | SCU_MODE_HIGHSPEEDSLEW_EN | SCU_MODE_INBUFF_EN | SCU_MODE_ZIF_DIS)**

**#define CLK_OUT (SCU_MODE_REPEATER | SCU_MODE_HIGHSPEEDSLEW_EN | SCU_MODE_INBUFF_EN | SCU_MODE_ZIF_DIS)**

**#define delay_us( x) Timer_DelayUs(x)**

**#define EMC_IO (SCU_MODE_REPEATER | SCU_MODE_HIGHSPEEDSLEW_EN |**

```
SCU_MODE_INBUFF_EN  | SCU_MODE_ZIF_DIS)

#define FAIL  0

#define GPIO_NOPULL  (SCU_MODE_INACT   | SCU_MODE_INBUFF_EN  )

#define GPIO_PDN  (SCU_MODE_PULLDOWN  | SCU_MODE_INBUFF_EN  )

#define GPIO_PUP  (SCU_MODE_PULLUP      | SCU_MODE_INBUFF_EN  )

#define HP_PTTN  3,1,GPIO_NOPULL,FUNC4

#define HP_RTSN  3,2,GPIO_NOPULL,FUNC4

#define HP_U2_RXD  7,2,UART_RX_TX,FUNC6

#define HP_U2_TXD  7,1,UART_RX_TX,FUNC6

#define I2S_RX_WS  3,1,M_I2S,FUNC1

#define I2S_RXCLK  0xF,4,M_I2S,FUNC7

#define I2S_RXD  3,2,M_I2S,FUNC1

#define I2S_TXCLK  4,7,M_I2S,FUNC7

#define I2S_TXD  7,2,M_I2S,FUNC2

#define I2S_WS  7,1,M_I2S,FUNC2

#define LCD_PINCONFIG  (SCU_MODE_INACT          | SCU_MODE_HIGHSPEEDSLEW_EN |
SCU_MODE_INBUFF_EN  | SCU_MODE_ZIF_DIS)

#define LED_D1  0x10

#define LED_D1_PIN  9

#define LED_D1_PORT  2
```

```
#define LED_D1_SCU_PIN  0

#define LED_D1_SCU_PORT  5

#define LED_D2  0x11

#define LED_D2_PIN  10

#define LED_D2_PORT  2

#define LED_D2_SCU_PIN  1

#define LED_D2_SCU_PORT  5

#define LED_D3  0x12

#define LED_D3_PIN  12

#define LED_D3_PORT  2

#define LED_D3_SCU_PIN  3

#define LED_D3_SCU_PORT  5

#define LED_D4  0x13

#define LED_D4_PIN  14

#define LED_D4_PORT  2

#define LED_D4_SCU_PIN  5

#define LED_D4_SCU_PORT  5

#define M_ETH  MD_PLN_FAST

#define M_GPIO  GPIO_NOPULL
```

```
#define M_I2S  SCU_PINIO_FAST

#define OK  1

#define S_ND  0xFF

#define SCU_BOOTEN1_1  4,9,M_GPIO,FUNC4

#define SCU_BOOTEN1_2  4,10,M_GPIO,FUNC4

#define SCU_BOOTEN2_1  6,7,M_GPIO,FUNC4

#define SCU_BOOTEN2_2  6,8,M_GPIO,FUNC4

#define SCU_DIO1  5,0,M_GPIO,FUNC0

#define SCU_DIO2  5,1,M_GPIO,FUNC0

#define SCU_DIO3  5,3,M_GPIO,FUNC0

#define SCU_DIO4  5,5,M_GPIO,FUNC0

#define SCU_ENET_MDC  7,7,M_ETH,FUNC6

#define SCU_ENET_MDIO  1,17,M_ETH,FUNC3

#define SCU_ENET_REF_CLK  1,19,M_ETH,FUNC0

#define SCU_ENET_RX_D  1,16,M_ETH,FUNC7

#define SCU_ENET_RXD0  1,15,M_ETH,FUNC3

#define SCU_ENET_RXD1  0,0,M_ETH,FUNC2

#define SCU_ENET_TX_EN  0,1,M_ETH,FUNC6

#define SCU_ENET_TXD0  1,18,M_ETH,FUNC3
```

```
#define SCU_ENET_TXD1  1,20,M_ETH,FUNC3

#define SCU_GPIO0  4,0,M_GPIO,FUNC0

#define SCU_GPIO1  4,1,M_GPIO,FUNC0

#define SCU_GPIO2  4,2,M_GPIO,FUNC0

#define SCU_GPIO3  4,3,M_GPIO,FUNC0

#define SCU_GPIO4  4,4,M_GPIO,FUNC0

#define SCU_GPIO5  4,5,M_GPIO,FUNC0

#define SCU_GPIO6  4,6,M_GPIO,FUNC0

#define SCU_GPIO7  3,5,M_GPIO,FUNC0

#define SCU_IO0  1,0,M_GPIO,FUNC0

#define SCU_IO1  6,2,M_GPIO,FUNC0

#define SCU_IO10  6,10,M_GPIO,FUNC0

#define SCU_IO11  6,11,M_GPIO,FUNC0

#define SCU_IO12  5,7,M_GPIO,FUNC0

#define SCU_IO13  2,2,M_GPIO,FUNC4

#define SCU_IO14  2,3,M_GPIO,FUNC4

#define SCU_IO15  2,5,M_GPIO,FUNC4

#define SCU_IO2  6,3,M_GPIO,FUNC0

#define SCU_IO3  6,5,M_GPIO,FUNC0
```

```
#define SCU_IO4  7,4,M_GPIO,FUNC0

#define SCU_IO5  7,5,M_GPIO,FUNC0

#define SCU_IO6  7,6,M_GPIO,FUNC0

#define SCU_IO7  7,0,M_GPIO,FUNC0

#define SCU_IO8  7,3,M_GPIO,FUNC0

#define SCU_IO9  6,9,M_GPIO,FUNC0

#define SCU_IRQN1  4,8,GPIO_PUP,FUNC4

#define SCU_IRQN2  5,6,GPIO_PUP,FUNC0

#define SD_CLK_SCU_PIN  CLK2

#define SD_CLK_SCU_PORT  CLK2

#define SD_CMD_SCU_PIN  6

#define SD_CMD_SCU_PORT  1

#define SD_DAT0_SCU_PIN  9

#define SD_DAT0_SCU_PORT  1

#define SD_DAT1_SCU_PIN  10

#define SD_DAT1_SCU_PORT  1

#define SD_DAT2_SCU_PIN  11

#define SD_DAT2_SCU_PORT  1

#define SD_DAT3_SCU_PIN  12
```

```
#define SD_DAT3_SCU_PORT  1

#define SET_SSP0  0x40

#define SET_SSP1  0x41

#define SSP0_MISO_SCU_PIN  7

#define SSP0_MISO_SCU_PORT  3

#define SSP0_MOSI_SCU_PIN  8

#define SSP0_MOSI_SCU_PORT  3

#define SSP0_SCK_SCU_PIN  0

#define SSP0_SCK_SCU_PORT  3

#define SSP0_SSEL_GPIO_MASK  (1 << SSP0_SSEL_GPIO_PIN)

#define SSP0_SSEL_GPIO_PIN  6

#define SSP0_SSEL_GPIO_PORT  0

#define SSP0_SSEL_SCU_PIN  6

#define SSP0_SSEL_SCU_PORT  3

#define SSP1_MISO_SCU_PIN  3

#define SSP1_MISO_SCU_PORT  1

#define SSP1_MOSI_SCU_PIN  4

#define SSP1_MOSI_SCU_PORT  1

#define SSP1_SCK_SCU_PIN  CLK0
```

```
#define SSP1_SCK_SCU_PORT  CLK0

#define SSP1_SSEL_GPIO_MASK  (1 << SSP1_SSEL_GPIO_PIN)

#define SSP1_SSEL_GPIO_PIN  8

#define SSP1_SSEL_GPIO_PORT  1

#define SSP1_SSEL_SCU_PIN  5

#define SSP1_SSEL_SCU_PORT  1

#define SSP_IO  (SCU_MODE_REPEATER  | SCU_MODE_HIGHSPEEDSLEW_EN |
SCU_MODE_INBUFF_EN  | SCU_MODE_ZIF_DIS)

#define U0_CTS_PIN  13

#define U0_CTS_PORT  2

#define U0_CTS_SCU_PIN  4

#define U0_CTS_SCU_PORT  5

#define U0_RTS_PIN  11

#define U0_RTS_PORT  2

#define U0_RTS_SCU_PIN  2

#define U0_RTS_SCU_PORT  5

#define U0_RXD_SCU_PIN  1

#define U0_RXD_SCU_PORT  2

#define U0_TXD_SCU_PIN  0

#define U0_TXD_SCU_PORT  2
```

**#define U1_CTS_SCU_PIN  4**

**#define U1_CTS_SCU_PORT  5**

**#define U1_RTS_SCU_PIN  2**

**#define U1_RTS_SCU_PORT  5**

**#define U1_RXD_SCU_PIN  14**

**#define U1_RXD_SCU_PORT  1**

**#define U1_TXD_SCU_PIN  13**

**#define U1_TXD_SCU_PORT  1**

**#define U2_RXD_SCU_PIN  11**

**#define U2_RXD_SCU_PORT  2**

**#define U2_TXD_SCU_PIN  10**

**#define U2_TXD_SCU_PORT  2**

**#define UART_RX_TX  (SCU_MODE_REPEATER | SCU_MODE_INBUFF_EN  )**

### 4.7.4 Function Documentation

**void Pe0003_BoardEthernetSet ()**

Initialise the Ethernet port pins.

**Returns:**
None

**void Pe0003_BoardGpioInit ()**

Initialise the GPIO pins used by the controllers.

**Returns:**

None

### void Pe0003_BoardHostPortUart2Init ()

Initialise the Host port a UART configuration.

**Returns:**

None

### void Pe0003_BoardI2sInit ()

Initialise the HostPort I2S pins.

**Returns:**

None

### void Pe0003_BoardInit ()

Configure all the peripheral pins of the PE0003 board and set the clock.

**Returns:**

None

**Note:**

Faster way to configure all peripheral pins

### void Pe0003_BoardLedInit ()

Initialise the LEDs.

**Returns:**

None

**Note:**

This An alternative method provides an alternative way to control initialise the leds LEDs. Use different to the one provider by the **gpio.h** Use it for fast tests

### void Pe0003_BoardLedSet (uint8_t *ledNumber*, uint8_t *state*)

Control a LED.

**Parameters:**

| | |
|---|---|
| *ledNumber* | - LED to control. LED_D1, LED_D2, LED_D3, LED_D4 |

| | |
|---|---|
| *state* | - 1 to switch on and 0 to switch off |

**Returns:**

None

**Note:**

This An alternative method provides an alternative way to control initialise the leds LEDs. Use different to the one provider by the **gpio.h** Use it for fast tests

### void Pe0003_BoardLedToggle (uint8_t *ledNumber*)

Toggle a LED state.

**Parameters:**

| | |
|---|---|
| *ledNumber* | - LED to toggle. LED_D1, LED_D2, LED_D3, LED_D4 |

**Returns:**

None

**Note:**

This An alternative method provides an alternative way to control initialise the leds LEDs. Use different to the one provider by the **gpio.h** Use it for fast tests

### void Pe0003_BoardSdCardInit ()

Initialise the pins for the SD Card controller.

**Returns:**

None

### void Pe0003_BoardSspInit (LPC_SSP_T * *pSSP*)

Initialise the SSP port pins, used for CBUS.

**Parameters:**

| | |
|---|---|
| *pSSP* | - Pointer to serial port. LPC_SSP0, LPC_SSP1, CBUS1, CBUS2 |

**Returns:**

None

**Note:**

The LPC_SSP0 and LPC_SSP2 are used by CBUS1 and CBUS2.

### void Pe0003_BoardUsartDeInit (LPC_USART_T * *pUART*)

Deinitialise UART/USART.

**Parameters:**

| | |
|---|---|
| *pUART* | - UART/USART to deinitialise. LPC_USART0, LPC_UART1, LPC_USART2 USB_FTDI, HOSTPORT_UART, DEBUG_UART |

**Returns:**

None

### void Pe0003_BoardUsartInit (LPC_USART_T * *pUART*)

Initialise the pins for an UART/USART.

**Parameters:**

| | |
|---|---|
| *pUART* | - UART/USART to configure. Values LPC_USART0, LPC_UART1, LPC_USART2, USB_FTDI, HOSTPORT_UART, DEBUG_UART |

**Returns:**

None

**Note:**

Check pe0003h for configurations
USB_FTDI uses LPC_UART1
HOSTPORT_UART uses LPC_USART2
DEBUG_UART uses LPC_USART2

### void Pe0003_DbgInit ()

Configure UART2 to use COM port for debugging. Note that: It is possible to use the NXP debugging framework from NXP with the proper if correctly configured. Other options for debugging include using the semihost configuration with printf, scanf functions and the LPCXpresso console.

**Returns:**

None

**Note:**

Method for debugging using USART2 and virtual COMPort

### uint8_t Pe0003_DbgUartGetChar ()

Debug port function. Get char.

**Returns:**

Char

### void Pe0003_DbgUartGetStr (void * *str*)

Debug port funtion. Get a string.

**Parameters:**

| | |
|---|---|
| *str* | - pointer to string |

**Returns:**

None

### void Pe0003_DbgUartPutChar (uint8_t *val*)

Debug port function put char.

**Parameters:**

| | |
|---|---|
| *val* | - char to write |

**Returns:**

Nothing

### void Pe0003_DbgUartPutStr (const void * *str*)

Debug port function put a string.

**Parameters:**

| | |
|---|---|
| *str* | - String to write |

**Returns:**

None

### void Pe0003_InitClock ()

Clock Initialisation. Set the main frequency to the maximum of 204MHz.

**Returns:**

None

### 4.8 inc/pe0003_config.h File Reference

#### 4.8.1 Macros

- #define **__LIB_VERSION_MAJOR** 1
- #define **__LIB_VERSION_MINOR** 0
- #define **CHIP_LPC43XX**
- #define **SDCARD_ENABLE**
  *Enables the SDCard peripheral.*
- #define **HOSTPORT_U2S_ENABLE**
  *Enables the HostPort with Uart configuration.*
- #define **TIMER0_ENABLE**
  *Enables Timer0.*
- #define **LEDS_ENABLE**
  *Enables an alternative way to control the LEDs.*
- #define **USB_INT**
- #define **GPIO_INT**
- #define **USB_FTDI** LPC_UART1
- #define **CBUS1** LPC_SSP0
- #define **CBUS2** LPC_SSP1
- #define **I2SP** LPC_I2S0
- #define **DEBUG_UART** LPC_USART2
- #define **HOSTPORT_UART** LPC_USART2
- #define **CRYSTAL_MAIN_FREQ_IN** 12000000
- #define **EXTERNAL_CLKIN_FREQ_IN** 0
- #define **MAX_CLOCK_FREQ** (204000000)
- #define **CLK_CORE** (204000000)
- #define **DEBUG_BAUDRATE** 115200
- #define **DEBUGINIT**()
- #define **DEBUGOUT**(...)
- #define **DEBUGSTR**(str)
- #define **DEBUGIN**() (int) **EOF**

#### 4.8.2 Macro Definition Documentation

**#define __LIB_VERSION_MAJOR 1**

**#define __LIB_VERSION_MINOR 0**

**#define CBUS1 LPC_SSP0**

CBus1 define SSP1

**#define CBUS2 LPC_SSP1**

CBus2 define SSP0

**#define CHIP_LPC43XX**

**#define CLK_CORE (204000000)**

Main clock frequency operation

**#define CRYSTAL_MAIN_FREQ_IN  12000000**

Crystal frequency into device

**#define DEBUG_BAUDRATE  115200**

Comment out DEBUG_ENABLE for IO support via the UART
// DEBUG_UART_ENABLE
Default configuration
Baudrate = 115200bps
8 data bit
1 Stop bit
None parity

**#define DEBUG_UART  LPC_USART2**

Debug COM Port

**#define DEBUGIN()  (int) EOF**

**#define DEBUGINIT()**

**#define DEBUGOUT(  ...)**

**#define DEBUGSTR( str)**

**#define EXTERNAL_CLKIN_FREQ_IN  0**

Frequency on external clock in pin - Unused

**#define GPIO_INT**

**#define HOSTPORT_U2S_ENABLE**

Enables the HostPort with Uart configuration.

**#define HOSTPORT_UART  LPC_USART2**

Host Port Uart

**#define I2SP  LPC_I2S0**

Host Port I2S

#### #define LEDS_ENABLE

Enables an alternative way to control the LEDs.

#### #define MAX_CLOCK_FREQ  (204000000)

Maximum CPU clock frequency

#### #define SDCARD_ENABLE

Enables the SDCard peripheral.

#### #define TIMER0_ENABLE

Enables Timer0.

#### #define USB_FTDI  LPC_UART1

Uart used for ftdi USB

#### #define USB_INT

## 4.9 inc/sdmmc.h File Reference

### 4.9.1 Macros

- #define **SDIO_TIMER** LPC_TIMER0

### 4.9.2 Functions

- int32_t **Chip_SDMMC_EraseBlocks** (LPC_SDMMC_T *pSDMMC, uint32_t start_block, uint32_t num_blocks)
- bool **Chip_SDMMC_EraseBusy** (LPC_SDMMC_T *pSDMMC)
- int32_t **Chip_SDMMC_SuperWriteInit** (LPC_SDMMC_T *pSDMMC, uint32_t start_block, uint32_t num_blocks)
- int32_t **Chip_SDMMC_SuperWriteInitNoDMA** (LPC_SDMMC_T *pSDMMC, uint32_t start_block, uint32_t num_blocks)
- uint32_t **Chip_SDMMC_SuperWrite** (LPC_SDMMC_T *pSDMMC, uint32_t *buffer, uint32_t size)
- void **Chip_SDMMC_SuperWriteStop** (LPC_SDMMC_T *pSDMMC)
- INLINE bool **Chip_SDMMC_SuperWriteDone** (LPC_SDMMC_T *pSDMMC)
- uint32_t **Chip_SDMMC_SuperWriteMaxFifoFullTime** (void)
- uint32_t **Chip_SDMMC_Acquire** (LPC_SDMMC_T *pSDMMC, mci_card_struct *pcardinfo)
- void **sdmmc_waitms** (uint32_t time)
- void **sdmmc_setup_wakeup** (void *bits)
- uint32_t **sdmmc_irq_driven_wait** (void)
- void **sdmmc_app_Init** ()
- uint32_t **sdmmc_MaxNumberBlocks** ()

### 4.9.3 Macro Definition Documentation

**#define SDIO_TIMER LPC_TIMER0**

### 4.9.4 Function Documentation

**uint32_t Chip_SDMMC_Acquire (LPC_SDMMC_T * *pSDMMC*, mci_card_struct * *pcardinfo*)**

**int32_t Chip_SDMMC_EraseBlocks (LPC_SDMMC_T * *pSDMMC*, uint32_t *start_block*, uint32_t *num_blocks*)**

**bool Chip_SDMMC_EraseBusy (LPC_SDMMC_T * *pSDMMC*)**

**uint32_t Chip_SDMMC_SuperWrite (LPC_SDMMC_T * *pSDMMC*, uint32_t * *buffer*, uint32_t *size*)**

**INLINE bool Chip_SDMMC_SuperWriteDone (LPC_SDMMC_T * *pSDMMC*)**

**int32_t Chip_SDMMC_SuperWriteInit (LPC_SDMMC_T * *pSDMMC*, uint32_t *start_block*, uint32_t *num_blocks*)**

**int32_t Chip_SDMMC_SuperWriteInitNoDMA (LPC_SDMMC_T * *pSDMMC*, uint32_t *start_block*,**

**uint32_t *num_blocks*)**

**uint32_t Chip_SDMMC_SuperWriteMaxFifoFullTime (void )**

**void Chip_SDMMC_SuperWriteStop (LPC_SDMMC_T * *pSDMMC*)**

**void sdmmc_app_Init ()**

**uint32_t sdmmc_irq_driven_wait (void )**

**uint32_t sdmmc_MaxNumberBlocks ()**

**void sdmmc_setup_wakeup (void * *bits*)**

**void sdmmc_waitms (uint32_t *time*)**

## 4.10 inc/timer.h File Reference

### 4.10.1 Functions

- void **Pe0003_TimerInit** ()
  *Initialise TIMER0 This timer uses interrupts and global variable usec for counting microseconds resolution +-100usec //TODO improve to a different version of timer.*
- uint32_t **Pe0003_TimerUsec** ()
  *Returns timer in usec.*
- void **Pe0003_TimerDelayUs** (uint32_t delayus)
  *Generates a delay in us.*
- void **Pe0003_ResetTimer** (void)
- void **DisableTimer0Int** (void)
  *Disable TIMER0 interruptss.*
- void **EnableTimer0Int** (void)
  *Enable TIMER0 interrupts.*
- void **Pe0003_Timer1Init** (void)
  *Timer1 initialise The period time is set to 10us.*
- void **Pe0003_Timer1SetIntUs** (uint32_t val)
  *Configure Timer1 to interrupt after timer period.*
- void **EnableTimer1Int** ()
  *Disable Timer1 interrupt.*
- void **DisableTimer1Int** ()
  *Enable Timer1 interrupt.*

### 4.10.2 Variables

- volatile uint32_t **usec**

### 4.10.3 Function Documentation

#### void DisableTimer0Int (void )

Disable TIMER0 interruptss.

**Returns:**
None

#### void DisableTimer1Int ()

Enable Timer1 interrupt.

#### void EnableTimer0Int (void )

Enable TIMER0 interrupts.

**Returns:**
None

### void EnableTimer1Int ()

Disable Timer1 interrupt.

### void Pe0003_ResetTimer (void )

Reset the timer

### void Pe0003_Timer1Init (void )

Timer1 initialise The period time is set to 10us.

TIMER1

### void Pe0003_Timer1SetIntUs (uint32_t *val*)

Configure Timer1 to interrupt after timer period.

**Parameters:**

| | |
|---|---|
| *val* | - Timer period in us. |

**Note:**

The interrupt handler Timer1_IRQnHandler must be implemented

```
    void TIMER1_IRQHandler()
{
    //DO YOUR STUFF

    Chip_TIMER_ClearMatch(LPC_TIMER1, 0);
    NVIC_ClearPendingIRQ(TIMER1_IRQn);
    NVIC_DisableIRQ(TIMER1_IRQ);

}
```

### void Pe0003_TimerDelayUs (uint32_t *delayus*)

Generates a delay in us.

**Parameters:**

| | |
|---|---|
| *delayus* | - Delay in microseconds |

**Returns:**

None

### void Pe0003_TimerInit ()

Initialise TIMER0 This timer uses interrupts and global variable usec for counting microseconds resolution +-100usec //TODO improve to a different version of timer.

TIMER0

**Returns:**

None

### uint32_t Pe0003_TimerUsec ()

Returns timer in usec.

Check source file for interrupts TIMER0_IRQHandler

**Returns:**

None

---

## 4.10.4 Variable Documentation

### volatile uint32_t usec

TWO TIMERS USED

TIMER0 - Implements a clock and it is initialised with **Pe0003_TimerInit()**

TIMER1 - Implements a timer and it is initialised with

### 4.11 src/sysinit.c File Reference

```
#include "pe0003.h"
```

#### 4.11.1 Functions

- void **SystemInit** (void)

#### 4.11.2 Variables

- const uint32_t **ExtRateIn** = 0
- const uint32_t **OscRateIn** = 12000000

#### 4.11.3 Function Documentation

**void SystemInit (void )**

#### 4.11.4 Variable Documentation

**const uint32_t ExtRateIn = 0**

**const uint32_t OscRateIn = 12000000**

## 4.12 src/wait.c File Reference

### 4.12.1 Macros

- #define **TIMEUNIT** 24

### 4.12.2 Functions

- void **timer_wait_us** (void *t, volatile int us)
- void **timer_wait_ms** (void *t, volatile int ms)

### 4.12.3 Macro Definition Documentation

**#define TIMEUNIT 24**

### 4.12.4 Function Documentation

**void timer_wait_ms (void * *t*, volatile int *ms*)**

**void timer_wait_us (void * *t*, volatile int *us*)**

| | | |
|---|---|---|
| **CML Microcircuits (UK) Ltd**<br>*COMMUNICATION SEMICONDUCTORS* | **CML Microcircuits (USA) Inc.**<br>*COMMUNICATION SEMICONDUCTORS* | **CML Microcircuits (Singapore) Pte Ltd**<br>*COMMUNICATION SEMICONDUCTORS* |
| **Tel:**<br>+44 (0)1621 875500<br>**Fax:**<br>+44 (0)1621 875600<br>**Sales:**<br>sales@cmlmicro.com<br><br>**Tech Support:**<br>techsupport@cmlmicro.com | **Tel:**<br>+1 336 744 5050<br>800 638 5577<br>**Fax:**<br>+1 336 744 5054<br>**Sales:**<br>us.sales@cmlmicro.com<br>**Tech Support:**<br>us.techsupport@cmlmicro.com | **Tel:**<br>+65 62 888129<br>**Fax:**<br>+65 62 888230<br>**Sales:**<br>sg.sales@cmlmicro.com<br><br>**Tech Support:**<br>sg.techsupport@cmlmicro.com |
| **- www.cmlmicro.com -** | | |