My Project

Generated by Doxygen 1.8.7

Thu Jul 24 2014 17:32:31

Contents

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Chapter 2

Class Index

2.1 Class List

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

Chapter 3

File Index

3.1 File List

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

Chapter 4

Module Documentation

4.1 NXP LPC1769 LPCXpresso board software API functions

Modules

- BOARD: NXP LPC1769 LPCXpresso board build options

Macros

- #define **BOARD_NXP_LPCXPRESSO_1769**
- #define **USE_RMII**
- #define LEDS_LED1 0x01
- #define **LEDS_LED2** 0x02
- #define **LEDS_LED3** 0x04
- #define **LEDS_LED4** 0x08
- #define **LEDS_NO_LEDS** 0x00
- #define BUTTONS_BUTTON1 0x01
- #define **NO_BUTTON_PRESSED** 0x00
- #define JOY_UP 0x01
- #define **JOY_DOWN** 0x02
- #define **JOY_LEFT** 0x04
- #define **JOY_RIGHT** 0x08
- #define **JOY_PRESS** 0x10
- #define MCB_17XX_AUDIO_MIC_SELECT 0x00
- #define **MCB_17XX_AUDIO_LINE_IN_SELECT** 0x00

Functions

- void Board_ENET_GetMacADDR (uint8_t *mcaddr)

  Initialize pin muxing for a UART.
- void Board_SPI_Init (bool isMaster)

  Sets up board specific I2S interface and UDA1380 CODEC.
- void Board_SPI_AssertSSEL (void)

  Assert SSEL pin.
- void Board_SPI_DeassertSSEL (void)

  De-assert SSEL pin.

4.1.1 Detailed Description

The board support software API functions provide some simple abstracted functions used across multiple LPCOpen board examples. See BOARD: Common board functions for the functions defined by this board support layer.

4.1.2 Macro Definition Documentation

4.1.2.1 #define BUTTONS_BUTTON1 0x01

Button defines

4.1.2.2 #define JOY_UP 0x01

Joystick defines

4.1.2.3 #define LEDS_LED1 0x01

LED defines

4.1.2.4 #define MCB_17XX_AUDIO_MIC_SELECT 0x00

Dummy audio input selection values enum

4.1.3 Function Documentation

4.1.3.1 void Board_ENET_GetMacADDR (uint8_t * *mcaddr*)

Initialize pin muxing for a UART.

Parameters

pUART	: Pointer to UART register block for UART pins to init

Returns

Nothing Returns the MAC address assigned to this board

Parameters

mcaddr	: Pointer to 6-byte character array to populate with MAC address

Returns

Nothing

Note

Returns the MAC address used by Ethernet

4.1.3.2 void Board_SPI_AssertSSEL (void)

Assert SSEL pin.

Returns

Nothing

4.1.3.3 void Board_SPI_DeassertSSEL (void)

De-assert SSEL pin.

Returns

Nothing

4.1.3.4 void Board_SPI_Init (bool *isMaster*)

Sets up board specific I2S interface and UDA1380 CODEC.

Parameters

pI2S	: I2S peripheral to use (Must be LPC_I2S)
micIn	: If 1 MIC will be used as input device, if 0 LINE_IN will be used as input to Audio Codec.

Returns

Nothing Initialize pin muxing for SSP interface

Parameters

pSSP	: Pointer to SSP interface to initialize

Returns

Nothing Initialize pin muxing for SPI interface

Parameters

isMaster	: true for master mode, false for slave mode

Returns

Nothing

4.2 BOARD: NXP LPC1769 LPCXpresso board build options

Macros

- #define DEBUG_ENABLE
- #define DEBUG_UART LPC_UART3

4.2.1 Detailed Description

This board has options that configure its operation at build-time.

4.2.2 Macro Definition Documentation

4.2.2.1 #define DEBUG_ENABLE

Define DEBUG_ENABLE to enable IO via the DEBUGSTR, DEBUGOUT, and DEBUGIN macros. If not defined, DEBUG∗ functions will be optimized out of the code at build time.

4.2.2.2 #define DEBUG_UART LPC_UART3

Define DEBUG_SEMIHOSTING along with DEBUG_ENABLE to enable IO support via semihosting. You may need to use a C library that supports semihosting with this option.Board UART used for debug output and input using the DEBUG∗ macros. This is also the port used for Board_UARTPutChar, Board_UARTGetChar, and Board_UART↩ PutSTR functions.

4.3 BOARD: Common board functions

Macros

- #define **DEBUGINIT**()
- #define **DEBUGOUT**(...)
- #define **DEBUGSTR**(str)
- #define **DEBUGIN**() (int) EOF

Typedefs

- typedef void(∗ p_msDelay_func_t)(uint32_t)

  Function prototype for a MS delay function. Board layers or example code may define this function as needed.

Functions

- void Board_SystemInit (void)

  Setup and initialize hardware prior to call to main()
- void Board_SetupMuxing (void)

  Setup pin multiplexer per board schematics.
- void Board_SetupClocking (void)

  Setup system clocking.
- void Board_SetupExtMemory (void)

  Setup external system memory.
- void Board_Init (void)

  Set up and initialize all required blocks and functions related to the board hardware.
- void Board_Debug_Init (void)

  Initializes board UART for output, required for printf redirection.
- void Board_UARTPutChar (char ch)

  Sends a single character on the UART, required for printf redirection.
- int Board_UARTGetChar (void)

  Get a single character from the UART, required for scanf input.
- void Board_UARTPutSTR (char ∗str)

  Prints a string to the UART.
- void Board_LED_Set (uint8_t LEDNumber, bool State)

  Sets the state of a board LED to on or off.
- bool Board_LED_Test (uint8_t LEDNumber)

  Returns the current state of a board LED.
- void Board_LED_Toggle (uint8_t LEDNumber)

  Toggles the current state of a board LED.
- void Board_SetLCDBacklight (uint8_t Intensity)

  Turn on Board LCD Backlight.

4.3.1 Detailed Description

This file contains common board definitions that are shared across boards and devices. All of these functions do not need to be implemented for a specific board, but if they are implemented, they should use this API standard.

4.3.2 Function Documentation

4.3.2.1 void Board_Debug_Init (void)

Initializes board UART for output, required for printf redirection.

Returns

None

4.3.2.2 void Board_Init (void)

Set up and initialize all required blocks and functions related to the board hardware.

Returns

None

4.3.2.3 void Board_LED_Set (uint8_t *LEDNumber,* bool *State*)

Sets the state of a board LED to on or off.

Parameters

LEDNumber	: LED number to set state for
State	: true for on, false for off

Returns

None

4.3.2.4 bool Board_LED_Test (uint8_t *LEDNumber*)

Returns the current state of a board LED.

Parameters

LEDNumber	: LED number to set state for

Returns

true if the LED is on, otherwise false

4.3.2.5 void Board_LED_Toggle (uint8_t *LEDNumber*)

Toggles the current state of a board LED.

Parameters

LEDNumber	: LED number to change state for

Returns

None

4.3.2.6 void Board_SetLCDBacklight (uint8_t *Intensity*)

Turn on Board LCD Backlight.

Parameters

Intensity	: Backlight intensity (0 = off, >=1 = on)

Returns

None

Note

On boards where a GPIO is used to control backlight on/off state, a '0' or '1' value will turn off or on the backlight. On some boards, a non-0 value will control backlight intensity via a PWN. For PWM systems, the intensity value is a percentage value between 0 and 100%.

4.3.2.7 void Board_SetupClocking (void)

Setup system clocking.

Returns

None

Note

This sets up board clocking.

4.3.2.8 void Board_SetupExtMemory (void)

Setup external system memory.

Returns

None

Note

This function is typically called after pin mux setup and clock setup and sets up any external memory needed by the system (DRAM, SRAM, etc.). Not all boards need this function.

4.3.2.9 void Board_SetupMuxing (void)

Setup pin multiplexer per board schematics.

Returns

None

Note

Board_SetupMuxing() should be called from SystemInit() prior to application main() is called. So that the PINs are set in proper state.

4.3.2.10 void Board_SystemInit (void)

Setup and initialize hardware prior to call to main()

Returns

None

Note

Board_SystemInit() is called prior to the application and sets up system clocking, memory, and any resources needed prior to the application starting.

4.3.2.11 int Board_UARTGetChar (void)

Get a single character from the UART, required for scanf input.

Returns

EOF if not character was received, or character value

4.3.2.12 void Board_UARTPutChar (char *ch*)

Sends a single character on the UART, required for printf redirection.

Parameters

ch	: character to send

Returns

None

4.3.2.13 void Board_UARTPutSTR (char * *str*)

Prints a string to the UART.

Parameters

str	: Terminated string to output

Returns

None

4.4 : Common Buttons functions

Functions

- unsigned int BUTTONS_Read (unsigned int mask)

  Read from GPIO ports for a single mask.
- void BUTTONS_Init (unsigned int pinMap)

  Initialize GIPO port for Button function.

4.4.1 Detailed Description

This file contains common buttons functions.

4.4.2 Function Documentation

4.4.2.1 void BUTTONS_Init (unsigned int *pinMap*)

Initialize GIPO port for Button function.

Parameters

pin	map of buttons

Returns

Nothing

Note

This use GPIO ports for initialization

4.4.2.2 unsigned int BUTTONS_Read (unsigned int *mask*)

Read from GPIO ports for a single mask.

Parameters

Mask	of pin map from buttons

Returns

State of buttons, 1 for clicked, 0 for not clicked

Note

This use GPIO ports for reading LPC175x/6x devices.

4.5 CHIP: LPC17xx/40xx support functions

Functions

- void SystemCoreClockUpdate (void)

  Update system core clock rate, should be called if the system has a clock rate change.
- void Chip_SystemInit (void)

  Set up and initialize hardware prior to call to main()
- void Chip_USB_Init (void)

  USB Pin and clock initialization Calling this function will initialize the USB pins and the clock.
- void Chip_SetupXtalClocking (void)

  Clock and PLL initialization based on the external oscillator.
- void Chip_SetupIrcClocking (void)

  Clock and PLL initialization based on the internal oscillator.

Variables

- uint32_t SystemCoreClock

  Current system clock rate, mainly used for sysTick.

4.5.1 Detailed Description

4.5.2 Function Documentation

4.5.2.1 void Chip_SetupIrcClocking (void)

Clock and PLL initialization based on the internal oscillator.

Returns

None

4.5.2.2 void Chip_SetupXtalClocking (void)

Clock and PLL initialization based on the external oscillator.

Returns

None

Note

This function assumes an external crystal oscillator frequency of 12MHz.

4.5.2.3 void Chip_SystemInit (void)

Set up and initialize hardware prior to call to main()

Returns

None

Note

Chip_SystemInit() is called prior to the application and sets up system clocking prior to the application starting.

4.5.2.4 void Chip_USB_Init (void)

USB Pin and clock initialization Calling this function will initialize the USB pins and the clock.

Returns

None

Note

This function will assume that the chip is clocked by an external crystal oscillator of frequency 12MHz and the Oscillator is running.

4.5.2.5 void SystemCoreClockUpdate (void)

Update system core clock rate, should be called if the system has a clock rate change.

Returns

None

4.6 CHIP: LPC175x/6x Peripheral addresses and register set declarations

Macros

- #define **LPC_GPIO0_BASE** 0x2009C000
- #define **LPC_GPIO1_BASE** 0x2009C020
- #define **LPC_GPIO2_BASE** 0x2009C040
- #define **LPC_GPIO3_BASE** 0x2009C060
- #define **LPC_GPIO4_BASE** 0x2009C080
- #define **LPC_WWDT_BASE** 0x40000000
- #define **LPC_TIMER0_BASE** 0x40004000
- #define **LPC_TIMER1_BASE** 0x40008000
- #define **LPC_UART0_BASE** 0x4000C000
- #define **LPC_UART1_BASE** 0x40010000
- #define **LPC_PWM1_BASE** 0x40018000
- #define **LPC_I2C0_BASE** 0x4001C000
- #define **LPC_SPI_BASE** 0x40020000
- #define **LPC_RTC_BASE** 0x40024000
- #define **LPC_REGFILE_BASE** 0x40024044
- #define **LPC_GPIOINT_BASE** 0x40028080
- #define **LPC_IOCON_BASE** 0x4002C000
- #define **LPC_SSP1_BASE** 0x40030000
- #define **LPC_ADC_BASE** 0x40034000
- #define **LPC_CANAF_RAM_BASE** 0x40038000
- #define **LPC_CANAF_BASE** 0x4003C000
- #define **LPC_CANCR_BASE** 0x40040000
- #define **LPC_CAN1_BASE** 0x40044000
- #define **LPC_CAN2_BASE** 0x40048000
- #define **LPC_I2C1_BASE** 0x4005C000
- #define **LPC_FMC_BASE** 0x40084000
- #define **LPC_SSP0_BASE** 0x40088000
- #define **LPC_DAC_BASE** 0x4008C000
- #define **LPC_TIMER2_BASE** 0x40090000
- #define **LPC_TIMER3_BASE** 0x40094000
- #define **LPC_UART2_BASE** 0x40098000
- #define **LPC_UART3_BASE** 0x4009C000
- #define **LPC_I2C2_BASE** 0x400A0000
- #define **LPC_I2S_BASE** 0x400A8000
- #define **LPC_RITIMER_BASE** 0x400B0000
- #define **LPC_MCPWM_BASE** 0x400B8000
- #define **LPC_QEI_BASE** 0x400BC000
- #define **LPC_SYSCTL_BASE** 0x400FC000
- #define **LPC_PMU_BASE** 0x400FC0C0
- #define **LPC_ENET_BASE** 0x50000000
- #define **LPC_GPDMA_BASE** 0x50004000
- #define **LPC_USB_BASE** 0x5000C000
- #define **LPC_PMU** ((LPC_PMU_T *) LPC_PMU_BASE)
- #define **LPC_GPDMA** ((LPC_GPDMA_T *) LPC_GPDMA_BASE)
- #define **LPC_USB** ((LPC_USB_T *) LPC_USB_BASE)
- #define **LPC_ETHERNET** ((LPC_ENET_T *) LPC_ENET_BASE)
- #define **LPC_GPIO** ((LPC_GPIO_T *) LPC_GPIO0_BASE)
- #define **LPC_GPIO1** ((LPC_GPIO_T *) LPC_GPIO1_BASE)
- #define **LPC_GPIO2** ((LPC_GPIO_T *) LPC_GPIO2_BASE)
- #define **LPC_GPIO3** ((LPC_GPIO_T *) LPC_GPIO3_BASE)

- #define **LPC_GPIO4** ((LPC_GPIO_T *) LPC_GPIO4_BASE)
- #define **LPC_GPIOINT** ((LPC_GPIOINT_T *) LPC_GPIOINT_BASE)
- #define **LPC_RTC** ((LPC_RTC_T *) LPC_RTC_BASE)
- #define **LPC_REGFILE** ((LPC_REGFILE_T *) LPC_REGFILE_BASE)
- #define **LPC_WWDT** ((LPC_WWDT_T *) LPC_WWDT_BASE)
- #define **LPC_UART0** ((LPC_USART_T *) LPC_UART0_BASE)
- #define **LPC_UART1** ((LPC_USART_T *) LPC_UART1_BASE)
- #define **LPC_UART2** ((LPC_USART_T *) LPC_UART2_BASE)
- #define **LPC_UART3** ((LPC_USART_T *) LPC_UART3_BASE)
- #define **LPC_SPI** ((LPC_SPI_T *) LPC_SPI_BASE)
- #define **LPC_SSP0** ((LPC_SSP_T *) LPC_SSP0_BASE)
- #define **LPC_SSP1** ((LPC_SSP_T *) LPC_SSP1_BASE)
- #define **LPC_TIMER0** ((LPC_TIMER_T *) LPC_TIMER0_BASE)
- #define **LPC_TIMER1** ((LPC_TIMER_T *) LPC_TIMER1_BASE)
- #define **LPC_TIMER2** ((LPC_TIMER_T *) LPC_TIMER2_BASE)
- #define **LPC_TIMER3** ((LPC_TIMER_T *) LPC_TIMER3_BASE)
- #define **LPC_MCPWM** ((LPC_MCPWM_T *) LPC_MCPWM_BASE)
- #define **LPC_I2C0** ((LPC_I2C_T *) LPC_I2C0_BASE)
- #define **LPC_I2C1** ((LPC_I2C_T *) LPC_I2C1_BASE)
- #define **LPC_I2C2** ((LPC_I2C_T *) LPC_I2C2_BASE)
- #define **LPC_I2S** ((LPC_I2S_T *) LPC_I2S_BASE)
- #define **LPC_QEI** ((LPC_QEI_T *) LPC_QEI_BASE)
- #define **LPC_DAC** ((LPC_DAC_T *) LPC_DAC_BASE)
- #define **LPC_ADC** ((LPC_ADC_T *) LPC_ADC_BASE)
- #define **LPC_IOCON** ((LPC_IOCON_T *) LPC_IOCON_BASE)
- #define **LPC_SYSCTL** ((LPC_SYSCTL_T *) LPC_SYSCTL_BASE)
- #define **LPC_SYSCON** ((LPC_SYSCTL_T *) LPC_SYSCTL_BASE) /* Alias for LPC_SYSCTL */
- #define **LPC_CANAF_RAM** ((LPC_CANAF_RAM_T *) LPC_CANAF_RAM_BASE)
- #define **LPC_CANAF** ((LPC_CANAF_T *) LPC_CANAF_BASE)
- #define **LPC_CANCR** ((LPC_CANCR_T *) LPC_CANCR_BASE)
- #define **LPC_CAN1** ((LPC_CAN_T *) LPC_CAN1_BASE)
- #define **LPC_CAN2** ((LPC_CAN_T *) LPC_CAN2_BASE)
- #define **LPC_RITIMER** ((LPC_RITIMER_T *) LPC_RITIMER_BASE)
- #define **LPC_FMC** ((LPC_FMC_T *) LPC_FMC_BASE)
- #define **UART_IRQHandler** UART0_IRQHandler
- #define **I2C_IRQHandler** I2C0_IRQHandler
- #define **SSP_IRQHandler** SSP0_IRQHandler

4.6.1 Detailed Description

4.7 CHIP: LPC17xx/40xx Clock Driver

Macros

- #define SYSCTL_OSCRANGE_15_25 (1 << 4)
- #define SYSCTL_OSCEC (1 << 5)
- #define SYSCTL_OSCSTAT (1 << 6)
- #define **SYSCTL_IRC_FREQ** (4000000)
- #define SYSCTL_PLL_ENABLE (1 << 0)
- #define SYSCTL_PLL_CONNECT (1 << 1)
- #define SYSCTL_PLL0STS_ENABLED (1 << 24)
- #define SYSCTL_PLL0STS_CONNECTED (1 << 25)
- #define SYSCTL_PLL0STS_LOCKED (1 << 26)
- #define SYSCTL_PLL1STS_ENABLED (1 << 8)
- #define SYSCTL_PLL1STS_CONNECTED (1 << 9)
- #define SYSCTL_PLL1STS_LOCKED (1 << 10)

Typedefs

- typedef enum CHIP_SYSCTL_CLOCK CHIP_SYSCTL_CLOCK_T
- typedef enum CHIP_SYSCTL_CCLKSRC CHIP_SYSCTL_CCLKSRC_T
- typedef enum CHIP_SYSCTL_PLLCLKSRC CHIP_SYSCTL_PLLCLKSRC_T

Enumerations

- enum CHIP_SYSCTL_CLOCK {
  SYSCTL_CLOCK_RSVD0, SYSCTL_CLOCK_TIMER0, SYSCTL_CLOCK_TIMER1, SYSCTL_CLOCK_U↩ ART0,
  SYSCTL_CLOCK_UART1, **SYSCTL_CLOCK_RSVD5**, SYSCTL_CLOCK_PWM1, SYSCTL_CLOCK_I2C0, SYSCTL_CLOCK_SPI, SYSCTL_CLOCK_RTC, SYSCTL_CLOCK_SSP1, **SYSCTL_CLOCK_RSVD11**, SYSCTL_CLOCK_ADC, SYSCTL_CLOCK_CAN1, SYSCTL_CLOCK_CAN2, SYSCTL_CLOCK_GPIO, SYSCTL_CLOCK_RIT, SYSCTL_CLOCK_MCPWM, SYSCTL_CLOCK_QEI, SYSCTL_CLOCK_I2C1, **SYSCTL_CLOCK_RSVD20**, SYSCTL_CLOCK_SSP0, SYSCTL_CLOCK_TIMER2, SYSCTL_CLOCK_TI↩ MER3,
  SYSCTL_CLOCK_UART2, SYSCTL_CLOCK_UART3, SYSCTL_CLOCK_I2C2, SYSCTL_CLOCK_I2S, **SYSCTL_CLOCK_RSVD28**, SYSCTL_CLOCK_GPDMA, SYSCTL_CLOCK_ENET, SYSCTL_CLOCK_U↩ SB,
  SYSCTL_CLOCK_RSVD32, **SYSCTL_CLOCK_RSVD33**, **SYSCTL_CLOCK_RSVD34**, **SYSCTL_CLOC↩ K_RSVD35** }
- enum CHIP_SYSCTL_CCLKSRC { SYSCTL_CCLKSRC_SYSCLK, SYSCTL_CCLKSRC_MAINPLL }
- enum CHIP_SYSCTL_PLLCLKSRC { SYSCTL_PLLCLKSRC_IRC, SYSCTL_PLLCLKSRC_MAINOSC, S↩ YSCTL_PLLCLKSRC_RTC, **SYSCTL_PLLCLKSRC_RESERVED2** }
- enum CHIP_SYSCTL_CLKDIV_T {
  SYSCTL_CLKDIV_4, SYSCTL_CLKDIV_1, SYSCTL_CLKDIV_2, SYSCTL_CLKDIV_8,
  SYSCTL_CLKDIV_6_CCAN = SYSCTL_CLKDIV_8 }
- enum CHIP_SYSCTL_PCLK_T {
  SYSCTL_PCLK_WDT, SYSCTL_PCLK_TIMER0, SYSCTL_PCLK_TIMER1, SYSCTL_PCLK_UART0, SYSCTL_PCLK_UART1, **SYSCTL_PCLK_RSVD5**, SYSCTL_PCLK_PWM1, SYSCTL_PCLK_I2C0, SYSCTL_PCLK_SPI, **SYSCTL_PCLK_RSVD9**, SYSCTL_PCLK_SSP1, SYSCTL_PCLK_DAC, SYSCTL_PCLK_ADC, SYSCTL_PCLK_CAN1, SYSCTL_PCLK_CAN2, SYSCTL_PCLK_ACF, SYSCTL_PCLK_QEI, SYSCTL_PCLK_GPIOINT, SYSCTL_PCLK_PCB, SYSCTL_PCLK_I2C1, **SYSCTL_PCLK_RSVD20**, SYSCTL_PCLK_SSP0, SYSCTL_PCLK_TIMER2, SYSCTL_PCLK_TIMER3, SYSCTL_PCLK_UART2, SYSCTL_PCLK_UART3, SYSCTL_PCLK_I2C2, SYSCTL_PCLK_I2S, **SYSCTL_PCLK_RSVD28**, SYSCTL_PCLK_RIT, SYSCTL_PCLK_SYSCON, SYSCTL_PCLK_MCPWM }

- enum CHIP_SYSCTL_CLKOUTSRC_T {
  SYSCTL_CLKOUTSRC_CPU, SYSCTL_CLKOUTSRC_MAINOSC, SYSCTL_CLKOUTSRC_IRC, SYSCTL_CLKOUTSRC_USB,
  SYSCTL_CLKOUTSRC_RTC, **SYSCTL_CLKOUTSRC_RESERVED1**, **SYSCTL_CLKOUTSRC_RESERVED2**, **SYSCTL_CLKOUTSRC_RESERVED3** }

Functions

- void Chip_Clock_EnablePLL (CHIP_SYSCTL_PLL_T PLLNum, uint32_t flags)

  Enables or connects a PLL.
- void Chip_Clock_DisablePLL (CHIP_SYSCTL_PLL_T PLLNum, uint32_t flags)

  Disables or disconnects a PLL.
- void Chip_Clock_SetupPLL (CHIP_SYSCTL_PLL_T PLLNum, uint32_t msel, uint32_t psel)

  Sets up a PLL.
- STATIC INLINE uint32_t Chip_Clock_GetPLLStatus (CHIP_SYSCTL_PLL_T PLLNum)

  Returns PLL status.
- STATIC INLINE bool Chip_Clock_IsMainPLLEnabled (void)

  Read PLL0 enable status.
- STATIC INLINE bool Chip_Clock_IsUSBPLLEnabled (void)

  Read PLL1 enable status.
- STATIC INLINE bool Chip_Clock_IsMainPLLLocked (void)

  Read PLL0 lock status.
- STATIC INLINE bool Chip_Clock_IsUSBPLLLocked (void)

  Read PLL1 lock status.
- STATIC INLINE bool Chip_Clock_IsMainPLLConnected (void)

  Read PLL0 connect status.
- STATIC INLINE bool Chip_Clock_IsUSBPLLConnected (void)

  Read PLL1 lock status.
- STATIC INLINE void Chip_Clock_EnableCrystal (void)

  Enables the external Crystal oscillator.
- STATIC INLINE bool Chip_Clock_IsCrystalEnabled (void)

  Checks if the external Crystal oscillator is enabled.
- STATIC INLINE void Chip_Clock_SetCrystalRangeHi (void)

  Sets the external crystal oscillator range to 15Mhz - 25MHz.
- STATIC INLINE void Chip_Clock_SetCrystalRangeLo (void)

  Sets the external crystal oscillator range to 1Mhz - 20MHz.
- STATIC INLINE void Chip_Clock_FeedPLL (CHIP_SYSCTL_PLL_T PLLNum)

  Feeds a PLL.
- void Chip_Clock_EnablePeriphClock (CHIP_SYSCTL_CLOCK_T clk)

  Enables power and clocking for a peripheral.
- void Chip_Clock_DisablePeriphClock (CHIP_SYSCTL_CLOCK_T clk)

  Disables power and clocking for a peripheral.
- bool Chip_Clock_IsPeripheralClockEnabled (CHIP_SYSCTL_CLOCK_T clk)

  Returns power enables state for a peripheral.
- void Chip_Clock_SetCPUClockSource (CHIP_SYSCTL_CCLKSRC_T src)

  Sets the current CPU clock source.
- CHIP_SYSCTL_CCLKSRC_T Chip_Clock_GetCPUClockSource (void)

  Returns the current CPU clock source.
- void Chip_Clock_SetCPUClockDiv (uint32_t div)

  Sets the CPU clock divider.
- uint32_t Chip_Clock_GetCPUClockDiv (void)

Gets the CPU clock divider.

- void Chip_Clock_SetUSBClockDiv (uint32_t div)

  Sets the USB clock divider.
- uint32_t Chip_Clock_GetUSBClockDiv (void)

  Gets the USB clock divider.
- STATIC INLINE void Chip_Clock_SetMainPLLSource (CHIP_SYSCTL_PLLCLKSRC_T src)

  Selects a input clock source for SYSCLK.
- STATIC INLINE CHIP_SYSCTL_PLLCLKSRC_T Chip_Clock_GetMainPLLSource (void)

  Returns the input clock source for SYSCLK.
- void Chip_Clock_SetPCLKDiv (CHIP_SYSCTL_PCLK_T clk, CHIP_SYSCTL_CLKDIV_T div)

  Selects a clock divider for a peripheral.
- uint32_t Chip_Clock_GetPCLKDiv (CHIP_SYSCTL_PCLK_T clk)

  Gets a clock divider for a peripheral.
- void Chip_Clock_SetCLKOUTSource (CHIP_SYSCTL_CLKOUTSRC_T src, uint32_t div)

  Selects a source clock and divider rate for the CLKOUT pin.
- STATIC INLINE void Chip_Clock_EnableCLKOUT (void)

  Enables the clock on the CLKOUT pin.
- STATIC INLINE void Chip_Clock_DisableCLKOUT (void)

  Disables the clock on the CLKOUT pin.
- STATIC INLINE bool Chip_Clock_IsCLKOUTEnabled (void)

  Returns the CLKOUT activity indication status.
- STATIC INLINE uint32_t Chip_Clock_GetMainOscRate (void)

  Returns the main oscillator clock rate.
- STATIC INLINE uint32_t Chip_Clock_GetIntOscRate (void)

  Returns the internal oscillator (IRC) clock rate.
- STATIC INLINE uint32_t Chip_Clock_GetRTCOscRate (void)

  Returns the RTC oscillator clock rate.
- uint32_t Chip_Clock_GetSYSCLKRate (void)

  Returns the current SYSCLK clock rate.
- STATIC INLINE uint32_t Chip_Clock_GetMainPLLInClockRate (void)

  Return Main PLL (PLL0) input clock rate.
- uint32_t Chip_Clock_GetMainPLLOutClockRate (void)

  Return PLL0 (Main PLL) output clock rate.
- STATIC INLINE uint32_t Chip_Clock_GetUSBPLLInClockRate (void)

  Return USB PLL input clock rate.
- uint32_t Chip_Clock_GetUSBPLLOutClockRate (void)

  Return USB PLL output clock rate.
- uint32_t Chip_Clock_GetMainClockRate (void)

  Return main clock rate.
- uint32_t Chip_Clock_GetSystemClockRate (void)

  Return system clock rate.
- uint32_t Chip_Clock_GetUSBClockRate (void)

  Gets the USB clock (USB_CLK) rate.
- uint32_t Chip_Clock_GetPeripheralClockRate (CHIP_SYSCTL_PCLK_T clk)

  Returns clock rate for a peripheral (from peripheral clock)
- STATIC INLINE uint32_t Chip_Clock_GetRTCClockRate (void)

  Returns clock rate for RTC.
- STATIC INLINE uint32_t Chip_Clock_GetENETClockRate (void)

  Returns clock rate for Ethernet.
- STATIC INLINE uint32_t Chip_Clock_GetGPDMAClockRate (void)

  Returns clock rate for GPDMA.

4.7.1 Detailed Description

4.7.2 Macro Definition Documentation

4.7.2.1 #define SYSCTL_OSCEC (1 << 5)

SCS register - main oscillator enable

4.7.2.2 #define SYSCTL_OSCRANGE_15_25 (1 << 4)

SCS register - main oscillator range 15 to 25MHz

4.7.2.3 #define SYSCTL_OSCSTAT (1 << 6)

SCS register - main oscillator is ready status Internal oscillator frequency

4.7.2.4 #define SYSCTL_PLL0STS_CONNECTED (1 << 25)

PLL0 connect flag

4.7.2.5 #define SYSCTL_PLL0STS_ENABLED (1 << 24)

PLL0 enable flag

4.7.2.6 #define SYSCTL_PLL0STS_LOCKED (1 << 26)

PLL0 connect flag

4.7.2.7 #define SYSCTL_PLL1STS_CONNECTED (1 << 9)

PLL1 connect flag

4.7.2.8 #define SYSCTL_PLL1STS_ENABLED (1 << 8)

PLL1 enable flag

4.7.2.9 #define SYSCTL_PLL1STS_LOCKED (1 << 10)

PLL1 connect flag

4.7.2.10 #define SYSCTL_PLL_CONNECT (1 << 1)

PLL connect flag only applies to 175x/6x

4.7.2.11 #define SYSCTL_PLL_ENABLE (1 << 0)

PLL enable flag

4.7.3 Typedef Documentation

4.7.3.1 typedef enum CHIP_SYSCTL_CCLKSRC CHIP_SYSCTL_CCLKSRC_T

Selectable CPU clock sources

4.7.3.2 typedef enum CHIP_SYSCTL_CLOCK CHIP_SYSCTL_CLOCK_T

Power control for peripherals

4.7.3.3 typedef enum CHIP_SYSCTL_PLLCLKSRC CHIP_SYSCTL_PLLCLKSRC_T

PLL source clocks

4.7.4 Enumeration Type Documentation

4.7.4.1 enum CHIP_SYSCTL_CCLKSRC

Selectable CPU clock sources

Enumerator

SYSCTL_CCLKSRC_SYSCLK Select Sysclk as the input to the CPU clock divider.

SYSCTL_CCLKSRC_MAINPLL Select the output of the Main PLL as the input to the CPU clock divider.

4.7.4.2 enum CHIP_SYSCTL_CLKDIV_T

Clock and power peripheral clock divider rates used with the Clock_CLKDIVSEL_T clock types (devices only)

Enumerator

SYSCTL_CLKDIV_4 Divider by 4

SYSCTL_CLKDIV_1 Divider by 1

SYSCTL_CLKDIV_2 Divider by 2

SYSCTL_CLKDIV_8 Divider by 8, not for use with CAN

SYSCTL_CLKDIV_6_CCAN Divider by 6, CAN only

4.7.4.3 enum CHIP_SYSCTL_CLKOUTSRC_T

Clock sources for the CLKOUT pin

Enumerator

SYSCTL_CLKOUTSRC_CPU CPU clock as CLKOUT source

SYSCTL_CLKOUTSRC_MAINOSC Main oscillator clock as CLKOUT source

SYSCTL_CLKOUTSRC_IRC IRC oscillator clock as CLKOUT source

SYSCTL_CLKOUTSRC_USB USB clock as CLKOUT source

SYSCTL_CLKOUTSRC_RTC RTC clock as CLKOUT source

4.7.4.4 enum CHIP_SYSCTL_CLOCK

Power control for peripherals

Enumerator

SYSCTL_CLOCK_TIMER0 Timer 0 clock
SYSCTL_CLOCK_TIMER1 Timer 1 clock
SYSCTL_CLOCK_UART0 UART 0 clock
SYSCTL_CLOCK_UART1 UART 1 clock
SYSCTL_CLOCK_PWM1 PWM1 clock
SYSCTL_CLOCK_I2C0 I2C0 clock
SYSCTL_CLOCK_SPI SPI clock
SYSCTL_CLOCK_RTC RTC clock
SYSCTL_CLOCK_SSP1 SSP1 clock
SYSCTL_CLOCK_ADC ADC clock
SYSCTL_CLOCK_CAN1 CAN1 clock
SYSCTL_CLOCK_CAN2 CAN2 clock
SYSCTL_CLOCK_GPIO GPIO clock
SYSCTL_CLOCK_RIT RIT clock
SYSCTL_CLOCK_MCPWM MCPWM clock
SYSCTL_CLOCK_QEI QEI clock
SYSCTL_CLOCK_I2C1 I2C1 clock
SYSCTL_CLOCK_SSP0 SSP0 clock
SYSCTL_CLOCK_TIMER2 Timer 2 clock
SYSCTL_CLOCK_TIMER3 Timer 3 clock
SYSCTL_CLOCK_UART2 UART 2 clock
SYSCTL_CLOCK_UART3 UART 3 clock
SYSCTL_CLOCK_I2C2 I2C2 clock
SYSCTL_CLOCK_I2S I2S clock
SYSCTL_CLOCK_GPDMA GP DMA clock
SYSCTL_CLOCK_ENET EMAC/Ethernet clock
SYSCTL_CLOCK_USB USB clock

4.7.4.5 enum CHIP_SYSCTL_PCLK_T

Peripheral clock selection for LPC175x/6x This is a list of clocks that can be divided on the 175x/6x

Enumerator

SYSCTL_PCLK_WDT Watchdog divider
SYSCTL_PCLK_TIMER0 Timer 0 divider
SYSCTL_PCLK_TIMER1 Timer 1 divider
SYSCTL_PCLK_UART0 UART 0 divider
SYSCTL_PCLK_UART1 UART 1 divider
SYSCTL_PCLK_PWM1 PWM 1 divider
SYSCTL_PCLK_I2C0 I2C 0 divider
SYSCTL_PCLK_SPI SPI divider

SYSCTL_PCLK_SSP1 SSP 1 divider
SYSCTL_PCLK_DAC DAC divider
SYSCTL_PCLK_ADC ADC divider
SYSCTL_PCLK_CAN1 CAN 1 divider
SYSCTL_PCLK_CAN2 CAN 2 divider
SYSCTL_PCLK_ACF ACF divider
SYSCTL_PCLK_QEI QEI divider
SYSCTL_PCLK_GPIOINT GPIOINT divider
SYSCTL_PCLK_PCB PCB divider
SYSCTL_PCLK_I2C1 I2C 1 divider
SYSCTL_PCLK_SSP0 SSP 0 divider
SYSCTL_PCLK_TIMER2 Timer 2 divider
SYSCTL_PCLK_TIMER3 Timer 3 divider
SYSCTL_PCLK_UART2 UART 2 divider
SYSCTL_PCLK_UART3 UART 3 divider
SYSCTL_PCLK_I2C2 I2C 2 divider
SYSCTL_PCLK_I2S I2S divider
SYSCTL_PCLK_RIT Repetitive timer divider
SYSCTL_PCLK_SYSCON SYSCON divider
SYSCTL_PCLK_MCPWM Motor control PWM divider

4.7.4.6 enum CHIP_SYSCTL_PLLCLKSRC

PLL source clocks

Enumerator

SYSCTL_PLLCLKSRC_IRC PLL is sourced from the internal oscillator (IRC)
SYSCTL_PLLCLKSRC_MAINOSC PLL is sourced from the main oscillator
SYSCTL_PLLCLKSRC_RTC PLL is sourced from the RTC oscillator

4.7.5 Function Documentation

4.7.5.1 STATIC INLINE void Chip_Clock_DisableCLKOUT (void)

Disables the clock on the CLKOUT pin.

Returns

Nothing

4.7.5.2 void Chip_Clock_DisablePeriphClock (CHIP_SYSCTL_CLOCK_T *clk*)

Disables power and clocking for a peripheral.

Parameters

clk	Clock to disable

Returns

Nothing

Note

Only peripheral clocks that are defined in the PCONP registers of the clock and power controller can be enabled and disabled with this function. Some clocks need to be disabled elsewhere (ie, USB) and will return false to indicate it can't be disabled with this function.

4.7.5.3 void Chip_Clock_DisablePLL (CHIP_SYSCTL_PLL_T *PLLNum,* uint32_t *flags*)

Disables or disconnects a PLL.

Parameters

PLLNum	PLL number
flags	SYSCTL_PLL_ENABLE or SYSCTL_PLL_CONNECT

Returns

Nothing

Note

This will also perform a PLL feed sequence. Connect only applies to the LPC175x/6x devices.

4.7.5.4 STATIC INLINE void Chip_Clock_EnableCLKOUT (void)

Enables the clock on the CLKOUT pin.

Returns

Nothing

4.7.5.5 STATIC INLINE void Chip_Clock_EnableCrystal (void)

Enables the external Crystal oscillator.

Returns

Nothing

4.7.5.6 void Chip_Clock_EnablePeriphClock (CHIP_SYSCTL_CLOCK_T *clk*)

Enables power and clocking for a peripheral.

Parameters

clk	Clock to enable

Returns

Nothing

Note

Only peripheral clocks that are defined in the PCONP registers of the clock and power controller can be enabled and disabled with this function. Some clocks need to be enabled elsewhere (ie, USB) and will return false to indicate it can't be enabled with this function.

4.7.5.7 void Chip_Clock_EnablePLL (CHIP_SYSCTL_PLL_T *PLLNum,* uint32_t *flags*)

Enables or connects a PLL.

Parameters

PLLNum	PLL number
flags	SYSCTL_PLL_ENABLE or SYSCTL_PLL_CONNECT

Returns

Nothing

Note

This will also perform a PLL feed sequence. Connect only applies to the LPC175x/6x devices.

4.7.5.8 STATIC INLINE void Chip_Clock_FeedPLL (CHIP_SYSCTL_PLL_T *PLLNum*)

Feeds a PLL.

Parameters

PLLNum	PLL number

Returns

Nothing

4.7.5.9 uint32_t Chip_Clock_GetCPUClockDiv (void)

Gets the CPU clock divider.

Returns

CPU clock divider, between 1 and divider max

Note

The maximum divider for the 175x/6x is 256. The maximum divider for the 177x/8x and 407x/8x is 32. Note on 175x/6x devices, the divided CPU clock rate is used as the input to the peripheral clock dividers, while 177x/8x and 407x/8x devices use the undivided CPU clock rate.

4.7.5.10 CHIP_SYSCTL_CCLKSRC_T Chip_Clock_GetCPUClockSource (void)

Returns the current CPU clock source.

Returns

CPU clock source

Note

On 177x/8x and 407x/8x devices, this is also the peripheral clock source.

4.7.5.11 STATIC INLINE uint32_t Chip_Clock_GetENETClockRate (void)

Returns clock rate for Ethernet.

Returns

Clock rate for the peripheral

4.7.5.12 STATIC INLINE uint32_t Chip_Clock_GetGPDMAClockRate (void)

Returns clock rate for GPDMA.

Returns

Clock rate for the peripheral

4.7.5.13 STATIC INLINE uint32_t Chip_Clock_GetIntOscRate (void)

Returns the internal oscillator (IRC) clock rate.

Returns

internal oscillator (IRC) clock rate

4.7.5.14 uint32_t Chip_Clock_GetMainClockRate (void)

Return main clock rate.

Returns

main clock rate

4.7.5.15 STATIC INLINE uint32_t Chip_Clock_GetMainOscRate (void)

Returns the main oscillator clock rate.

Returns

main oscillator clock rate

4.7.5.16 STATIC INLINE uint32_t Chip_Clock_GetMainPLLInClockRate (void)

Return Main PLL (PLL0) input clock rate.

Returns

PLL0 input clock rate

4.7.5.17 uint32_t Chip_Clock_GetMainPLLOutClockRate (void)

Return PLL0 (Main PLL) output clock rate.

Returns

PLL0 output clock rate

4.7.5.18 STATIC INLINE CHIP_SYSCTL_PLLCLKSRC_T Chip_Clock_GetMainPLLSource (void)

Returns the input clock source for SYSCLK.

Returns

input clock source for SYSCLK

4.7.5.19 uint32_t Chip_Clock_GetPCLKDiv (CHIP_SYSCTL_PCLK_T *clk*)

Gets a clock divider for a peripheral.

Parameters

clk	Clock to set divider for

Returns

The divider for the clock

Note

Selects the divider for a peripheral. A peripheral clock is generated from the CPU clock divided by its peripheral clock divider. Only peripheral clocks that are defined in the PCLKSEL registers of the clock and power controller can used this function. (LPC175X/6X only)

4.7.5.20 uint32_t Chip_Clock_GetPeripheralClockRate (CHIP_SYSCTL_PCLK_T *clk*)

Returns clock rate for a peripheral (from peripheral clock)

Parameters

clk	Clock to get rate of

Returns

Clock rate for the peripheral

Note

This covers most common peripheral clocks, but not every clock in the system. LPC177x/8x and LPC407x/8x devices use the same clock for all periphreals, while the LPC175x/6x have unique dividers (except to RTC) that may alter the peripheral clock rate.

4.7.5.21 STATIC INLINE uint32_t Chip_Clock_GetPLLStatus (CHIP_SYSCTL_PLL_T *PLLNum*)

Returns PLL status.

Parameters

PLLNum	PLL number

Returns

Current enabled flags, Or'ed SYSCTL_PLLSTS_∗ states

Note

Note flag positions for PLL0 and PLL1 differ on the LPC175x/6x devices.

4.7.5.22 STATIC INLINE uint32_t Chip_Clock_GetRTCClockRate (void)

Returns clock rate for RTC.

Returns

Clock rate for the peripheral

4.7.5.23 STATIC INLINE uint32_t Chip_Clock_GetRTCOscRate (void)

Returns the RTC oscillator clock rate.

Returns

RTC oscillator clock rate

4.7.5.24 uint32_t Chip_Clock_GetSYSCLKRate (void)

Returns the current SYSCLK clock rate.

Returns

SYSCLK clock rate

Note

SYSCLK is used for sourcing PLL0, SPIFI FLASH, the USB clock divider, and the CPU clock divider.

4.7.5.25 uint32_t Chip_Clock_GetSystemClockRate (void)

Return system clock rate.

Returns

system clock rate

4.7.5.26 uint32_t Chip_Clock_GetUSBClockDiv (void)

Gets the USB clock divider.

Returns

USB clock divider

Note

Divider values are between 1 and 32 (16 max for 175x/6x)

4.7.5.27 uint32_t Chip_Clock_GetUSBClockRate (void)

Gets the USB clock (USB_CLK) rate.

Returns

USB clock (USB_CLK) clock rate

Note

The clock source and divider are used to generate the USB clock rate.

4.7.5.28 STATIC INLINE uint32_t Chip_Clock_GetUSBPLLInClockRate (void)

Return USB PLL input clock rate.

Returns

USB PLL input clock rate

4.7.5.29 uint32_t Chip_Clock_GetUSBPLLOutClockRate (void)

Return USB PLL output clock rate.

Returns

USB PLL output clock rate

4.7.5.30 STATIC INLINE bool Chip_Clock_IsCLKOUTEnabled (void)

Returns the CLKOUT activity indication status.

Returns

true if CLKOUT is enabled, false if disabled and stopped

Note

CLKOUT activity indication. Reads as true when CLKOUT is enabled. Read as false when CLKOUT has been disabled via the CLKOUT_EN bit and the clock has completed being stopped.

4.7.5.31 STATIC INLINE bool Chip_Clock_IsCrystalEnabled (void)

Checks if the external Crystal oscillator is enabled.

Returns

true if enabled, false otherwise

4.7.5.32 STATIC INLINE bool Chip_Clock_IsMainPLLConnected (void)

Read PLL0 connect status.

Returns

true of the PLL0 is connected. false if not connected

4.7.5.33 STATIC INLINE bool Chip_Clock_IsMainPLLEnabled (void)

Read PLL0 enable status.

Returns

true of the PLL0 is enabled. false if not enabled

4.7.5.34 STATIC INLINE bool Chip_Clock_IsMainPLLLocked (void)

Read PLL0 lock status.

Returns

true of the PLL0 is locked. false if not locked

4.7.5.35 bool Chip_Clock_IsPeripheralClockEnabled (CHIP_SYSCTL_CLOCK_T *clk*)

Returns power enables state for a peripheral.

Parameters

clk	Clock to check

Returns

true if the clock is enabled, false if disabled

4.7.5.36 STATIC INLINE bool Chip_Clock_IsUSBPLLConnected (void)

Read PLL1 lock status.

Returns

true of the PLL1 is connected. false if not connected

4.7.5.37 STATIC INLINE bool Chip_Clock_IsUSBPLLEnabled (void)

Read PLL1 enable status.

Returns

true of the PLL1 is enabled. false if not enabled

4.7.5.38 STATIC INLINE bool Chip_Clock_IsUSBPLLLocked (void)

Read PLL1 lock status.

Returns

true of the PLL1 is locked. false if not locked

4.7.5.39 void Chip_Clock_SetCLKOUTSource (CHIP_SYSCTL_CLKOUTSRC_T *src,* uint32_t *div*)

Selects a source clock and divider rate for the CLKOUT pin.

Parameters

src	source selected
div	Divider for the clock source on CLKOUT, 1 to 16

Returns

Nothing

Note

This function will disable the CLKOUT signal if its enabled. Use Chip_Clock_EnableCLKOUT to re-enable CLKOUT after a call to this function.

4.7.5.40 void Chip_Clock_SetCPUClockDiv (uint32_t *div*)

Sets the CPU clock divider.

Parameters

div	CPU clock divider, between 1 and divider max

Returns

Nothing

Note

The maximum divider for the 175x/6x is 256. The maximum divider for the 177x/8x and 407x/8x is 32. Note on 175x/6x devices, the divided CPU clock rate is used as the input to the peripheral clock dividers, while 177x/8x and 407x/8x devices use the undivided CPU clock rate.

4.7.5.41 void Chip_Clock_SetCPUClockSource (CHIP_SYSCTL_CCLKSRC_T *src*)

Sets the current CPU clock source.

Parameters

src	Source selected

Returns

Nothing

Note

When setting the clock source to the PLL, it should be enabled and locked.

4.7.5.42 STATIC INLINE void Chip_Clock_SetCrystalRangeHi (void)

Sets the external crystal oscillator range to 15Mhz - 25MHz.

Returns

Nothing

4.7.5.43 STATIC INLINE void Chip_Clock_SetCrystalRangeLo (void)

Sets the external crystal oscillator range to 1Mhz - 20MHz.

Returns

Nothing

4.7.5.44 STATIC INLINE void Chip_Clock_SetMainPLLSource (CHIP_SYSCTL_PLLCLKSRC_T *src*)

Selects a input clock source for SYSCLK.

Parameters

src	input clock source for SYSCLK

Returns

Nothing

Note

SYSCLK is used for sourcing PLL0, SPIFI FLASH, the USB clock divider, and the CPU clock divider.

4.7.5.45 void Chip_Clock_SetPCLKDiv (CHIP_SYSCTL_PCLK_T *clk,* CHIP_SYSCTL_CLKDIV_T *div*)

Selects a clock divider for a peripheral.

Parameters

clk	Clock to set divider for
div	Divider for the clock

Returns

Nothing

Note

Selects the divider for a peripheral. A peripheral clock is generated from the CPU clock divided by its peripheral clock divider. Only peripheral clocks that are defined in the PCLKSEL registers of the clock and power controller can used this function. (LPC175X/6X only)

4.7.5.46 void Chip_Clock_SetupPLL (CHIP_SYSCTL_PLL_T *PLLNum,* uint32_t *msel,* uint32_t *psel*)

Sets up a PLL.

Parameters

PLLNum	PLL number
msel	PLL Multiplier value (Must be pre-decremented)
psel	PLL Divider value (Must be pre-decremented)

Note

See the User Manual for limitations on these values for stable PLL operation. Be careful with these values - they must be safe values for the msl, nsel, and psel registers so must be already decremented by 1 or the the correct value for psel (0 = div by 1, 1 = div by 2, etc.).

Returns

Nothing

4.7.5.47 void Chip_Clock_SetUSBClockDiv (uint32_t *div*)

Sets the USB clock divider.

Parameters

div	USB clock divider to generate 48MHz from USB source clock

Returns

Nothing

Note

Divider values are between 1 and 32 (16 max for 175x/6x)

4.8 CHIP: LPC175x/6x CMSIS include file

Modules

- CHIP_175X_6X: LPC175x/6x peripheral interrupt numbers
- CHIP: LPC175x/6x Cortex CMSIS definitions

4.8.1 Detailed Description

4.9 CHIP_175X_6X: LPC175x/6x peripheral interrupt numbers

Enumerations

- enum LPC175X_6X_IRQn_Type {
  Reset_IRQn = -15, NonMaskableInt_IRQn = -14, HardFault_IRQn = -13, MemoryManagement_IRQn = -12,
  BusFault_IRQn = -11, UsageFault_IRQn = -10, SVCall_IRQn = -5, DebugMonitor_IRQn = -4,
  PendSV_IRQn = -2, SysTick_IRQn = -1, WDT_IRQn = 0, TIMER0_IRQn = 1,
  TIMER1_IRQn = 2, TIMER2_IRQn = 3, TIMER3_IRQn = 4, UART0_IRQn = 5,
  UART_IRQn = UART0_IRQn, UART1_IRQn = 6, UART2_IRQn = 7, UART3_IRQn = 8,
  PWM1_IRQn = 9, I2C0_IRQn = 10, I2C_IRQn = I2C0_IRQn, I2C1_IRQn = 11,
  I2C2_IRQn = 12, SPI_IRQn = 13, SSP0_IRQn = 14, SSP_IRQn = SSP0_IRQn,
  SSP1_IRQn = 15, PLL0_IRQn = 16, RTC_IRQn = 17, EINT0_IRQn = 18,
  EINT1_IRQn = 19, EINT2_IRQn = 20, EINT3_IRQn = 21, ADC_IRQn = 22,
  BOD_IRQn = 23, USB_IRQn = 24, CAN_IRQn = 25, DMA_IRQn = 26,
  I2S_IRQn = 27, ETHERNET_IRQn = 28, RITIMER_IRQn = 29, MCPWM_IRQn = 30,
  QEI_IRQn = 31, PLL1_IRQn = 32, USBActivity_IRQn = 33, CANActivity_IRQn = 34 }

4.9.1 Detailed Description

4.9.2 Enumeration Type Documentation

4.9.2.1 enum LPC175X_6X_IRQn_Type

Enumerator

Reset_IRQn 1 Reset Vector, invoked on Power up and warm reset

NonMaskableInt_IRQn 2 Non maskable Interrupt, cannot be stopped or preempted

HardFault_IRQn 3 Hard Fault, all classes of Fault

MemoryManagement_IRQn 4 Memory Management, MPU mismatch, including Access Violation and No Match

BusFault_IRQn 5 Bus Fault, Pre-Fetch-, Memory Access Fault, other address/memory related Fault

UsageFault_IRQn 6 Usage Fault, i.e. Undef Instruction, Illegal State Transition

SVCall_IRQn 11 System Service Call via SVC instruction

DebugMonitor_IRQn 12 CDebug Monitor

PendSV_IRQn 14 Pendable request for system service

SysTick_IRQn 15 System Tick Interrupt

WDT_IRQn Watchdog Timer Interrupt

TIMER0_IRQn Timer0 Interrupt

TIMER1_IRQn Timer1 Interrupt

TIMER2_IRQn Timer2 Interrupt

TIMER3_IRQn Timer3 Interrupt

UART0_IRQn UART0 Interrupt

UART_IRQn Alias for UART0 Interrupt

UART1_IRQn UART1 Interrupt

UART2_IRQn UART2 Interrupt

UART3_IRQn UART3 Interrupt

PWM1_IRQn PWM1 Interrupt

I2C0_IRQn I2C0 Interrupt

I2C_IRQn Alias for I2C0 Interrupt

I2C1_IRQn I2C1 Interrupt

I2C2_IRQn I2C2 Interrupt

SPI_IRQn SPI Interrupt

SSP0_IRQn SSP0 Interrupt

SSP_IRQn Alias for SSP0 Interrupt

SSP1_IRQn SSP1 Interrupt

PLL0_IRQn PLL0 Lock (Main PLL) Interrupt

RTC_IRQn Real Time Clock and event recorder Interrupt

EINT0_IRQn External Interrupt 0 Interrupt

EINT1_IRQn External Interrupt 1 Interrupt

EINT2_IRQn External Interrupt 2 Interrupt

EINT3_IRQn External Interrupt 3 Interrupt

ADC_IRQn A/D Converter Interrupt

BOD_IRQn Brown-Out Detect Interrupt

USB_IRQn USB Interrupt

CAN_IRQn CAN Interrupt

DMA_IRQn General Purpose DMA Interrupt

I2S_IRQn I2S Interrupt

ETHERNET_IRQn Ethernet Interrupt

RITIMER_IRQn Repetitive Interrupt Interrupt

MCPWM_IRQn Motor Control PWM Interrupt

QEI_IRQn Quadrature Encoder Interface Interrupt

PLL1_IRQn PLL1 Lock (USB PLL) Interrupt

USBActivity_IRQn USB Activity interrupt

CANActivity_IRQn CAN Activity interrupt

4.10 CHIP: LPC175x/6x Cortex CMSIS definitions

Macros

- #define **__CM3_REV** 0x0200
- #define __MPU_PRESENT 1
- #define __NVIC_PRIO_BITS 5
- #define __Vendor_SysTickConfig 0
- #define __FPU_PRESENT 0

4.10.1 Detailed Description

4.10.2 Macro Definition Documentation

4.10.2.1 #define __FPU_PRESENT 0

FPU present or not

4.10.2.2 #define __MPU_PRESENT 1

MPU present or not

4.10.2.3 #define __NVIC_PRIO_BITS 5

Number of Bits used for Priority Levels

4.10.2.4 #define __Vendor_SysTickConfig 0

Set to 1 if different SysTick Config is used

4.11 : Common Ethernet functions

Functions

- void InitDescriptors (void)

  Initialize MAC descriptors for simple packet receive/transmit.
- void * ENET_RXGet (int32_t *bytes)

  Get the pointer to the Rx buffer storing new received frame.
- void ENET_RXBuffClaim (void)

  Release Rx Buffer.
- int ENET_checkBuffer (void)

  Debugging purposes.
- void * ENET_TXBuffGet (void)

  Get Tx Buffer for the next transmission.
- void ENET_TXQueue (int32_t bytes)

  Queue a new frame for transmission.
- bool ENET_IsTXFinish (void)

  Check if tranmission finished.

4.11.1 Detailed Description

This file contains common Ethernet functions.

4.11.2 Function Documentation

4.11.2.1 int ENET_checkBuffer (void)

Debugging purposes.

Parameters

Nothing	

Returns

Nothing

4.11.2.2 bool ENET_IsTXFinish (void)

Check if tranmission finished.

Parameters

Nothing	

Returns

Nothing

4.11.2.3 void ENET_RXBuffClaim (void)

Release Rx Buffer.

Parameters

Nothing	

Returns

Nothing

4.11.2.4 void* ENET_RXGet (int32_t * *bytes*)

Get the pointer to the Rx buffer storing new received frame.

Parameters

Nothing	

Returns

Nothing

4.11.2.5 void* ENET_TXBuffGet (void)

Get Tx Buffer for the next transmission.

Parameters

Nothing	

Returns

Nothing

4.11.2.6 void ENET_TXQueue (int32_t *bytes*)

Queue a new frame for transmission.

Parameters

Nothing	

Returns

Nothing

4.11.2.7 void InitDescriptors (void)

Initialize MAC descriptors for simple packet receive/transmit.

Parameters

Nothing	

Returns

Nothing

4.12 CHIP: LPC17xx/40xx Ethernet driver (2)

Classes

- struct ENET_MAC_T

  Ethernet MAC register block structure.
- struct ENET_TRANSFER_INFO_T

  Ethernet Transfer register Block Structure.
- struct ENET_CONTROL_T

  Ethernet Control register block structure.
- struct ENET_RXFILTER_T

  Ethernet Receive Filter register block structure.
- struct ENET_MODULE_CTRL_T

  Ethernet Module Control register block structure.
- struct LPC_ENET_T

  Ethernet register block structure.
- struct ENET_RXDESC_T

  RX Descriptor structure.
- struct ENET_RXSTAT_T

  RX Status structure.
- struct ENET_TXDESC_T

  TX Descriptor structure.
- struct ENET_TXSTAT_T

  TX Status structure.

Macros

- #define ENET_MAC1_MASK 0xcf1f
- #define ENET_MAC1_RXENABLE 0x00000001
- #define ENET_MAC1_PARF 0x00000002
- #define ENET_MAC1_RXFLOWCTRL 0x00000004
- #define ENET_MAC1_TXFLOWCTRL 0x00000008
- #define ENET_MAC1_LOOPBACK 0x00000010
- #define ENET_MAC1_RESETTX 0x00000100
- #define ENET_MAC1_RESETMCSTX 0x00000200
- #define ENET_MAC1_RESETRX 0x00000400
- #define ENET_MAC1_RESETMCSRX 0x00000800
- #define ENET_MAC1_SIMRESET 0x00004000
- #define ENET_MAC1_SOFTRESET 0x00008000
- #define ENET_MAC2_MASK 0x73ff
- #define ENET_MAC2_FULLDUPLEX 0x00000001
- #define ENET_MAC2_FLC 0x00000002
- #define ENET_MAC2_HFEN 0x00000004
- #define ENET_MAC2_DELAYEDCRC 0x00000008
- #define ENET_MAC2_CRCEN 0x00000010
- #define ENET_MAC2_PADCRCEN 0x00000020
- #define ENET_MAC2_VLANPADEN 0x00000040
- #define ENET_MAC2_AUTODETPADEN 0x00000080
- #define ENET_MAC2_PPENF 0x00000100
- #define ENET_MAC2_LPENF 0x00000200
- #define ENET_MAC2_NOBACKOFF 0x00001000
- #define ENET_MAC2_BP_NOBACKOFF 0x00002000

- #define ENET_MAC2_EXCESSDEFER 0x00004000
- #define ENET_IPGT_BTOBINTEGAP(n) ((n) & 0x7F)
- #define ENET_IPGT_FULLDUPLEX (ENET_IPGT_BTOBINTEGAP(0x15))
- #define ENET_IPGT_HALFDUPLEX (ENET_IPGT_BTOBINTEGAP(0x12))
- #define ENET_IPGR_NBTOBINTEGAP2(n) ((n) & 0x7F)
- #define ENET_IPGR_P2_DEF (ENET_IPGR_NBTOBINTEGAP2(0x12))
- #define ENET_IPGR_NBTOBINTEGAP1(n) (((n) & 0x7F) << 8)
- #define ENET_IPGR_P1_DEF ENET_IPGR_NBTOBINTEGAP1(0x0C)
- #define ENET_CLRT_RETRANSMAX(n) ((n) & 0x0F)
- #define ENET_CLRT_COLLWIN(n) (((n) & 0x3F) << 8)
- #define ENET_CLRT_DEF ((ENET_CLRT_RETRANSMAX(0x0F)) | (ENET_CLRT_COLLWIN(0x37)))
- #define ENET_MAXF_MAXFLEN(n) ((n) & 0xFFFF)
- #define **ENET_MAXF_MAXFLEN_DEF** (0x600)
- #define ENET_SUPP_100Mbps_SPEED 0x00000100
- #define ENET_TEST_SCPQ 0x00000001
- #define ENET_TEST_TESTPAUSE 0x00000002
- #define ENET_TEST_TESTBP 0x00000004
- #define ENET_MCFG_SCANINC 0x00000001
- #define ENET_MCFG_SUPPPREAMBLE 0x00000002
- #define ENET_MCFG_CLOCKSEL(n) (((n) & 0x0F) << 2)
- #define ENET_MCFG_RES_MII 0x00008000
- #define ENET_MCFG_RESETMIIMGMT 2500000UL
- #define ENET_MCMD_READ 0x00000001
- #define ENET_MCMD_SCAN 0x00000002
- #define ENET_MII_WR_TOUT 0x00050000
- #define ENET_MII_RD_TOUT 0x00050000
- #define ENET_MADR_REGADDR(n) ((n) & 0x1F)
- #define ENET_MADR_PHYADDR(n) (((n) & 0x1F) << 8)
- #define ENET_MWTD_DATA(n) ((n) & 0xFFFF)
- #define ENET_MRDD_DATA(n) ((n) & 0xFFFF)

  MII Management Read Data Register bit definitions.
- #define ENET_MIND_BUSY 0x00000001
- #define ENET_MIND_SCANNING 0x00000002
- #define ENET_MIND_NOTVALID 0x00000004
- #define ENET_MIND_MIILINKFAIL 0x00000008
- #define ENET_COMMAND_RXENABLE 0x00000001
- #define ENET_COMMAND_TXENABLE 0x00000002
- #define ENET_COMMAND_REGRESET 0x00000008
- #define ENET_COMMAND_TXRESET 0x00000010
- #define ENET_COMMAND_RXRESET 0x00000020
- #define ENET_COMMAND_PASSRUNTFRAME 0x00000040
- #define ENET_COMMAND_PASSRXFILTER 0x00000080
- #define ENET_COMMAND_TXFLOWCONTROL 0x00000100
- #define ENET_COMMAND_RMII 0x00000200
- #define ENET_COMMAND_FULLDUPLEX 0x00000400
- #define ENET_STATUS_RXSTATUS 0x00000001
- #define ENET_STATUS_TXSTATUS 0x00000002
- #define ENET_TSV0_CRCERR 0x00000001
- #define ENET_TSV0_LCE 0x00000002
- #define ENET_TSV0_LOR 0x00000004
- #define ENET_TSV0_DONE 0x00000008
- #define ENET_TSV0_MULTICAST 0x00000010
- #define ENET_TSV0_BROADCAST 0x00000020
- #define ENET_TSV0_PACKETDEFER 0x00000040

- #define ENET_TSV0_EXDF 0x00000080
- #define ENET_TSV0_EXCOL 0x00000100
- #define ENET_TSV0_LCOL 0x00000200
- #define ENET_TSV0_GIANT 0x00000400
- #define ENET_TSV0_UNDERRUN 0x00000800
- #define ENET_TSV0_TOTALBYTES 0x0FFFF000
- #define ENET_TSV0_CONTROLFRAME 0x10000000
- #define ENET_TSV0_PAUSE 0x20000000
- #define ENET_TSV0_BACKPRESSURE 0x40000000
- #define ENET_TSV0_VLAN 0x80000000
- #define ENET_TSV1_TBC 0x0000FFFF
- #define ENET_TSV1_TCC 0x000F0000
- #define ENET_RSV_RBC 0x0000FFFF
- #define ENET_RSV_PPI 0x00010000
- #define ENET_RSV_RXDVSEEN 0x00020000
- #define ENET_RSV_CESEEN 0x00040000
- #define ENET_RSV_RCV 0x00080000
- #define ENET_RSV_CRCERR 0x00100000
- #define ENET_RSV_LCERR 0x00200000
- #define ENET_RSV_LOR 0x00400000
- #define ENET_RSV_ROK 0x00800000
- #define ENET_RSV_MULTICAST 0x01000000
- #define ENET_RSV_BROADCAST 0x02000000
- #define ENET_RSV_DRIBBLENIBBLE 0x04000000
- #define ENET_RSV_CONTROLFRAME 0x08000000
- #define ENET_RSV_PAUSE 0x10000000
- #define ENET_RSV_UO 0x20000000
- #define ENET_RSV_VLAN 0x40000000
- #define ENET_FLOWCONTROLCOUNTER_MC(n) ((n) & 0xFFFF)
- #define ENET_FLOWCONTROLCOUNTER_PT(n) (((n) & 0xFFFF) << 16)
- #define ENET_FLOWCONTROLSTATUS_MCC(n) ((n) & 0xFFFF)
- #define ENET_RXFILTERCTRL_AUE 0x00000001
- #define ENET_RXFILTERCTRL_ABE 0x00000002
- #define ENET_RXFILTERCTRL_AME 0x00000004
- #define ENET_RXFILTERCTRL_AUHE 0x00000008
- #define ENET_RXFILTERCTRL_AMHE 0x00000010
- #define ENET_RXFILTERCTRL_APE 0x00000020
- #define ENET_RXFILTERCTRL_MPEW 0x00001000
- #define ENET_RXFILTERCTRL_RFEW 0x00002000
- #define ENET_RXFILTERWOLSTATUS_AUW 0x00000001
- #define ENET_RXFILTERWOLSTATUS_ABW 0x00000002
- #define ENET_RXFILTERWOLSTATUS_AMW 0x00000004
- #define ENET_RXFILTERWOLSTATUS_AUHW 0x00000008
- #define ENET_RXFILTERWOLSTATUS_AMHW 0x00000010
- #define ENET_RXFILTERWOLSTATUS_APW 0x00000020
- #define ENET_RXFILTERWOLSTATUS_RFW 0x00000080
- #define ENET_RXFILTERWOLSTATUS_MPW 0x00000100
- #define ENET_RXFILTERWOLSTATUS_BITMASK 0x01BF
- #define ENET_INT_RXOVERRUN 0x00000001
- #define ENET_INT_RXERROR 0x00000002
- #define ENET_INT_RXFINISHED 0x00000004
- #define ENET_INT_RXDONE 0x00000008
- #define ENET_INT_TXUNDERRUN 0x00000010
- #define ENET_INT_TXERROR 0x00000020
- #define ENET_INT_TXFINISHED 0x00000040

- #define ENET_INT_TXDONE 0x00000080
- #define ENET_INT_SOFT 0x00001000
- #define ENET_INT_WAKEUP 0x00002000
- #define ENET_POWERDOWN_PD 0x80000000
- #define ENET_RCTRL_SIZE(n) (((n) - 1) & 0x7FF)

  RX Descriptor Control structure type definition.
- #define ENET_RCTRL_INT 0x80000000
- #define ENET_RHASH_SA 0x000001FF
- #define ENET_RHASH_DA 0x001FF000
- #define ENET_RINFO_SIZE(n) (((n) & 0x7FF) + 1)
- #define ENET_RINFO_CTRL_FRAME 0x00040000
- #define ENET_RINFO_VLAN 0x00080000
- #define ENET_RINFO_FAIL_FILT 0x00100000
- #define ENET_RINFO_MCAST 0x00200000
- #define ENET_RINFO_BCAST 0x00400000
- #define ENET_RINFO_CRC_ERR 0x00800000
- #define ENET_RINFO_SYM_ERR 0x01000000
- #define ENET_RINFO_LEN_ERR 0x02000000
- #define ENET_RINFO_RANGE_ERR 0x04000000
- #define ENET_RINFO_ALIGN_ERR 0x08000000
- #define ENET_RINFO_OVERRUN 0x10000000
- #define ENET_RINFO_NO_DESCR 0x20000000
- #define ENET_RINFO_LAST_FLAG 0x40000000
- #define ENET_RINFO_ERR 0x80000000
- #define ENET_RINFO_ERR_MASK
- #define ENET_TCTRL_SIZE(n) (((n) - 1) & 0x7FF)
- #define ENET_TCTRL_OVERRIDE 0x04000000
- #define ENET_TCTRL_HUGE 0x08000000
- #define ENET_TCTRL_PAD 0x10000000
- #define ENET_TCTRL_CRC 0x20000000
- #define ENET_TCTRL_LAST 0x40000000
- #define ENET_TCTRL_INT 0x80000000
- #define ENET_TINFO_COL_CNT 0x01E00000
- #define ENET_TINFO_DEFER 0x02000000
- #define ENET_TINFO_EXCESS_DEF 0x04000000
- #define ENET_TINFO_EXCESS_COL 0x08000000
- #define ENET_TINFO_LATE_COL 0x10000000
- #define ENET_TINFO_UNDERRUN 0x20000000
- #define ENET_TINFO_NO_DESCR 0x40000000
- #define ENET_TINFO_ERR 0x80000000
- #define ENET_ETH_MAX_FLEN (1536)

  Maximum size of an ethernet buffer.

Enumerations

- enum ENET_BUFF_STATUS_T { **ENET_BUFF_EMPTY**, **ENET_BUFF_PARTIAL_FULL**, **ENET_BUFF_F↩ ULL** }

  ENET Buffer status definition.

Functions

- STATIC INLINE void Chip_ENET_Reset (LPC_ENET_T *pENET)

  Resets the ethernet interface.
- STATIC INLINE void Chip_ENET_SetADDR (LPC_ENET_T *pENET, const uint8_t *macAddr)

  Sets the address of the interface.
- void Chip_ENET_SetupMII (LPC_ENET_T *pENET, uint32_t div, uint8_t addr)

  Sets up the PHY link clock divider and PHY address.
- void Chip_ENET_StartMIIWrite (LPC_ENET_T *pENET, uint8_t reg, uint16_t data)

  Starts a PHY write via the MII.
- void Chip_ENET_StartMIIRead (LPC_ENET_T *pENET, uint8_t reg)

  Starts a PHY read via the MII.
- STATIC INLINE bool Chip_ENET_IsMIIBusy (LPC_ENET_T *pENET)

  Returns MII link (PHY) busy status.
- uint16_t Chip_ENET_ReadMIIData (LPC_ENET_T *pENET)

  Returns the value read from the PHY.
- STATIC INLINE void Chip_ENET_TXEnable (LPC_ENET_T *pENET)

  Enables ethernet transmit.
- STATIC INLINE void Chip_ENET_TXDisable (LPC_ENET_T *pENET)

  Disables ethernet transmit.
- STATIC INLINE void Chip_ENET_RXEnable (LPC_ENET_T *pENET)

  Enables ethernet packet reception.
- STATIC INLINE void Chip_ENET_RXDisable (LPC_ENET_T *pENET)

  Disables ethernet packet reception.
- STATIC INLINE void Chip_ENET_ResetTXLogic (LPC_ENET_T *pENET)

  Reset Tx Logic.
- STATIC INLINE void Chip_ENET_ResetRXLogic (LPC_ENET_T *pENET)

  Reset Rx Logic.
- STATIC INLINE void Chip_ENET_EnableRXFilter (LPC_ENET_T *pENET, uint32_t mask)

  Enable Rx Filter.
- STATIC INLINE void Chip_ENET_DisableRXFilter (LPC_ENET_T *pENET, uint32_t mask)

  Disable Rx Filter.
- void Chip_ENET_SetFullDuplex (LPC_ENET_T *pENET)

  Sets full duplex operation for the interface.
- void Chip_ENET_SetHalfDuplex (LPC_ENET_T *pENET)

  Sets half duplex operation for the interface.
- STATIC INLINE void Chip_ENET_Set100Mbps (LPC_ENET_T *pENET)

  Selects 100Mbps for the current speed.
- STATIC INLINE void Chip_ENET_Set10Mbps (LPC_ENET_T *pENET)

  Selects 10Mbps for the current speed.
- void Chip_ENET_InitTxDescriptors (LPC_ENET_T *pENET, ENET_TXDESC_T *pDescs, ENET_TXSTAT_T *pStatus, uint32_t descNum)

  Configures the initial ethernet transmit descriptors.
- void Chip_ENET_InitRxDescriptors (LPC_ENET_T *pENET, ENET_RXDESC_T *pDescs, ENET_RXSTAT_T *pStatus, uint32_t descNum)

  Configures the initial ethernet receive descriptors.
- STATIC INLINE uint16_t Chip_ENET_GetTXProduceIndex (LPC_ENET_T *pENET)

  Get the current Tx Produce Descriptor Index.
- STATIC INLINE uint16_t Chip_ENET_GetTXConsumeIndex (LPC_ENET_T *pENET)

  Get the current Tx Consume Descriptor Index.
- STATIC INLINE uint16_t Chip_ENET_GetRXProduceIndex (LPC_ENET_T *pENET)

Get the current Rx Produce Descriptor Index.

- STATIC INLINE uint16_t Chip_ENET_GetRXConsumeIndex (LPC_ENET_T *pENET)

  Get the current Rx Consume Descriptor Index.

- ENET_BUFF_STATUS_T Chip_ENET_GetBufferStatus (LPC_ENET_T *pENET, uint16_t produceIndex, uint16_t consumeIndex, uint16_t buffSize)

  Get the buffer status with the current Produce Index and Consume Index.

- uint32_t Chip_ENET_GetFillDescNum (LPC_ENET_T *pENET, uint16_t produceIndex, uint16_t consume↩ Index, uint16_t buffSize)

  Get the number of descriptors filled.

- STATIC INLINE uint32_t Chip_ENET_GetFreeDescNum (LPC_ENET_T *pENET, uint16_t produceIndex, uint16_t consumeIndex, uint16_t buffSize)

  Get the number of free descriptors.

- STATIC INLINE bool Chip_ENET_IsTxFull (LPC_ENET_T *pENET)

  Check if Tx buffer is full.

- STATIC INLINE bool Chip_ENET_IsRxEmpty (LPC_ENET_T *pENET)

  Check if Rx buffer is empty.

- uint16_t Chip_ENET_IncTXProduceIndex (LPC_ENET_T *pENET)

  Increase the current Tx Produce Descriptor Index.

- uint16_t Chip_ENET_IncRXConsumeIndex (LPC_ENET_T *pENET)

  Increase the current Rx Consume Descriptor Index.

- STATIC INLINE void Chip_ENET_EnableInt (LPC_ENET_T *pENET, uint32_t mask)

  Enable ENET interrupts.

- STATIC INLINE void Chip_ENET_DisableInt (LPC_ENET_T *pENET, uint32_t mask)

  Disable ENET interrupts.

- STATIC INLINE uint32_t Chip_ENET_GetIntStatus (LPC_ENET_T *pENET)

  Get the interrupt status.

- STATIC INLINE void Chip_ENET_ClearIntStatus (LPC_ENET_T *pENET, uint32_t mask)

  Clear the interrupt status.

- void Chip_ENET_Init (LPC_ENET_T *pENET, bool useRMII)

  Initialize ethernet interface.

- void **Chip_ENET_Setup** (LPC_ENET_T *pENET, bool useRMII)
- void Chip_ENET_DeInit (LPC_ENET_T *pENET)

  De-initialize the ethernet interface.

- uint32_t Chip_ENET_FindMIIDiv (LPC_ENET_T *pENET, uint32_t clockRate)

  Find the divider index for a desired MII clock rate.

4.12.1 Detailed Description

4.12.2 Macro Definition Documentation

4.12.2.1 #define ENET_CLRT_COLLWIN(*n*) (((n) & 0x3F) << 8)

Programmable field representing the slot time or collision window during which collisions occur in properly configured networks

4.12.2.2 #define ENET_CLRT_DEF ((ENET_CLRT_RETRANSMAX(0x0F)) | (ENET_CLRT_COLLWIN(0x37)))

Default value for Collision Window / Retry register

4.12.2.3 #define ENET_CLRT_RETRANSMAX(*n*) ((n) & 0x0F)

Programmable field specifying the number of retransmission attempts following a collision before aborting the packet due to excessive collisions

4.12.2.4 #define ENET_COMMAND_FULLDUPLEX 0x00000400

Full Duplex

4.12.2.5 #define ENET_COMMAND_PASSRUNTFRAME 0x00000040

Pass Runt Frames

4.12.2.6 #define ENET_COMMAND_PASSRXFILTER 0x00000080

Pass RX Filter

4.12.2.7 #define ENET_COMMAND_REGRESET 0x00000008

Reset Host Registers

4.12.2.8 #define ENET_COMMAND_RMII 0x00000200

Reduced MII Interface

4.12.2.9 #define ENET_COMMAND_RXENABLE 0x00000001

Enable Receive

4.12.2.10 #define ENET_COMMAND_RXRESET 0x00000020

Reset Receive Datapath

4.12.2.11 #define ENET_COMMAND_TXENABLE 0x00000002

Enable Transmit

4.12.2.12 #define ENET_COMMAND_TXFLOWCONTROL 0x00000100

TX Flow Control

4.12.2.13 #define ENET_COMMAND_TXRESET 0x00000010

Reset Transmit Datapath

4.12.2.14 #define ENET_FLOWCONTROLCOUNTER_MC(*n*) ((n) & 0xFFFF)

Mirror Counter

4.12.2.15 #define ENET_FLOWCONTROLCOUNTER_PT(n) (((n) & 0xFFFF) << 16)

Pause Timer

4.12.2.16 #define ENET_FLOWCONTROLSTATUS_MCC(n) ((n) & 0xFFFF)

Mirror Counter Current

4.12.2.17 #define ENET_INT_RXDONE 0x00000008

Receive Done

4.12.2.18 #define ENET_INT_RXERROR 0x00000002

Receive Error

4.12.2.19 #define ENET_INT_RXFINISHED 0x00000004

RX Finished Process Descriptors

4.12.2.20 #define ENET_INT_RXOVERRUN 0x00000001

Overrun Error in RX Queue

4.12.2.21 #define ENET_INT_SOFT 0x00001000

Software Triggered Interrupt

4.12.2.22 #define ENET_INT_TXDONE 0x00000080

Transmit Done

4.12.2.23 #define ENET_INT_TXERROR 0x00000020

Transmit Error

4.12.2.24 #define ENET_INT_TXFINISHED 0x00000040

TX Finished Process Descriptors

4.12.2.25 #define ENET_INT_TXUNDERRUN 0x00000010

Transmit Underrun

4.12.2.26 #define ENET_INT_WAKEUP 0x00002000

Wakeup Event Interrupt

4.12.2.27 #define ENET_IPGR_NBTOBINTEGAP1(n) (((n) & 0x7F) << 8)

Programmable field representing the optional carrierSense window referenced in IEEE 802.3/4.2.3.2.1 'Carrier Deference'

4.12.2.28 #define ENET_IPGR_NBTOBINTEGAP2(n) ((n) & 0x7F)

Programmable field representing the Non-Back-to-Back Inter-Packet-Gap

4.12.2.29 #define ENET_IPGR_P1_DEF ENET_IPGR_NBTOBINTEGAP1(0x0C)

Recommended value for Programmable field representing the Non-Back-to-Back Inter-Packet-Gap Part 2

4.12.2.30 #define ENET_IPGR_P2_DEF (ENET_IPGR_NBTOBINTEGAP2(0x12))

Recommended value for Programmable field representing the Non-Back-to-Back Inter-Packet-Gap Part 1

4.12.2.31 #define ENET_IPGT_BTOBINTEGAP(n) ((n) & 0x7F)

Programmable field representing the nibble time offset of the minimum possible period between the end of any transmitted packet to the beginning of the next

4.12.2.32 #define ENET_IPGT_FULLDUPLEX (ENET_IPGT_BTOBINTEGAP(0x15))

Recommended value for Full Duplex of Programmable field representing the nibble time offset of the minimum possible period between the end of any transmitted packet to the beginning of the next

4.12.2.33 #define ENET_IPGT_HALFDUPLEX (ENET_IPGT_BTOBINTEGAP(0x12))

Recommended value for Half Duplex of Programmable field representing the nibble time offset of the minimum possible period between the end of any transmitted packet to the beginning of the next

4.12.2.34 #define ENET_MAC1_LOOPBACK 0x00000010

Loop Back Mode

4.12.2.35 #define ENET_MAC1_MASK 0xcf1f

MAC1 register mask

4.12.2.36 #define ENET_MAC1_PARF 0x00000002

Pass All Receive Frames

4.12.2.37 #define ENET_MAC1_RESETMCSRX 0x00000800

Reset MAC RX Control Sublayer

4.12.2.38 #define ENET_MAC1_RESETMCSTX 0x00000200

Reset MAC TX Control Sublayer

4.12.2.39 #define ENET_MAC1_RESETRX 0x00000400

Reset RX Logic

4.12.2.40 #define ENET_MAC1_RESETTX 0x00000100

Reset TX Logic

4.12.2.41 #define ENET_MAC1_RXENABLE 0x00000001

Receive Enable

4.12.2.42 #define ENET_MAC1_RXFLOWCTRL 0x00000004

RX Flow Control

4.12.2.43 #define ENET_MAC1_SIMRESET 0x00004000

Simulation Reset

4.12.2.44 #define ENET_MAC1_SOFTRESET 0x00008000

Soft Reset MAC

4.12.2.45 #define ENET_MAC1_TXFLOWCTRL 0x00000008

TX Flow Control

4.12.2.46 #define ENET_MAC2_AUTODETPADEN 0x00000080

Auto Detect Pad Enable

4.12.2.47 #define ENET_MAC2_BP_NOBACKOFF 0x00002000

Backoff Presurre / No Backoff

4.12.2.48 #define ENET_MAC2_CRCEN 0x00000010

Append CRC to every Frame

4.12.2.49 #define ENET_MAC2_DELAYEDCRC 0x00000008

Delayed CRC Mode

4.12.2.50 #define ENET_MAC2_EXCESSDEFER 0x00004000

Excess Defer

4.12.2.51 #define ENET_MAC2_FLC 0x00000002

Frame Length Checking

4.12.2.52 #define ENET_MAC2_FULLDUPLEX 0x00000001

Full-Duplex Mode

4.12.2.53 #define ENET_MAC2_HFEN 0x00000004

Huge Frame Enable

4.12.2.54 #define ENET_MAC2_LPENF 0x00000200

Long Preamble Enforcement

4.12.2.55 #define ENET_MAC2_MASK 0x73ff

MAC2 register mask

4.12.2.56 #define ENET_MAC2_NOBACKOFF 0x00001000

No Backoff Algorithm

4.12.2.57 #define ENET_MAC2_PADCRCEN 0x00000020

Pad all Short Frames

4.12.2.58 #define ENET_MAC2_PPENF 0x00000100

Pure Preamble Enforcement

4.12.2.59 #define ENET_MAC2_VLANPADEN 0x00000040

VLAN Pad Enable

4.12.2.60 #define ENET_MADR_PHYADDR(*n*) (((n) & 0x1F) << 8)

PHY Address Field

4.12.2.61 #define ENET_MADR_REGADDR(*n*) ((n) & 0x1F)

MII Register Address field

4.12.2.62 #define ENET_MAXF_MAXFLEN(n) ((n) & 0xFFFF)

Represents a maximum receive frame of 1536 octets

4.12.2.63 #define ENET_MCFG_CLOCKSEL(n) (((n) & 0x0F) << 2)

Clock Select Field

4.12.2.64 #define ENET_MCFG_RES_MII 0x00008000

Reset MII Management Hardware

4.12.2.65 #define ENET_MCFG_RESETMIIMGMT 2500000UL

MII Clock max

4.12.2.66 #define ENET_MCFG_SCANINC 0x00000001

Scan Increment PHY Address

4.12.2.67 #define ENET_MCFG_SUPPPREAMBLE 0x00000002

Suppress Preamble

4.12.2.68 #define ENET_MCMD_READ 0x00000001

MII Read

4.12.2.69 #define ENET_MCMD_SCAN 0x00000002

MII Scan continuously

4.12.2.70 #define ENET_MII_RD_TOUT 0x00050000

MII Read timeout count

4.12.2.71 #define ENET_MII_WR_TOUT 0x00050000

MII Write timeout count

4.12.2.72 #define ENET_MIND_BUSY 0x00000001

MII is Busy

4.12.2.73 #define ENET_MIND_MIILINKFAIL 0x00000008

MII Link Failed

4.12.2.74 #define ENET_MIND_NOTVALID 0x00000004

MII Read Data not valid

4.12.2.75 #define ENET_MIND_SCANNING 0x00000002

MII Scanning in Progress

4.12.2.76 #define ENET_MRDD_DATA(*n*) ((n) & 0xFFFF)

MII Management Read Data Register bit definitions.
Data field for MMI Management Read Data register

4.12.2.77 #define ENET_MWTD_DATA(*n*) ((n) & 0xFFFF)

Data field for MMI Management Write Data register

4.12.2.78 #define ENET_POWERDOWN_PD 0x80000000

Power Down MAC

4.12.2.79 #define ENET_RCTRL_INT 0x80000000

Generate RxDone Interrupt

4.12.2.80 #define ENET_RCTRL_SIZE(*n*) (((n) - 1) & 0x7FF)

RX Descriptor Control structure type definition.
Buffer size field

4.12.2.81 #define ENET_RHASH_DA 0x001FF000

Hash CRC for Destination Address

4.12.2.82 #define ENET_RHASH_SA 0x000001FF

Hash CRC for Source Address

4.12.2.83 #define ENET_RINFO_ALIGN_ERR 0x08000000

Alignment Error

4.12.2.84 #define ENET_RINFO_BCAST 0x00400000

Broadcast Frame

4.12.2.85 #define ENET_RINFO_CRC_ERR 0x00800000

CRC Error in Frame

4.12.2.86 #define ENET_RINFO_CTRL_FRAME 0x00040000

Control Frame

4.12.2.87 #define ENET_RINFO_ERR 0x80000000

Error Occured (OR of all errors)

4.12.2.88 #define ENET_RINFO_ERR_MASK

Value:

```
(ENET_RINFO_FAIL_FILT | ENET_RINFO_CRC_ERR    |
      ENET_RINFO_SYM_ERR |  \
                              ENET_RINFO_LEN_ERR    |
      ENET_RINFO_ALIGN_ERR | ENET_RINFO_OVERRUN)
```

RX Error status mask

4.12.2.89 #define ENET_RINFO_FAIL_FILT 0x00100000

RX Filter Failed

4.12.2.90 #define ENET_RINFO_LAST_FLAG 0x40000000

Last Fragment in Frame

4.12.2.91 #define ENET_RINFO_LEN_ERR 0x02000000

Length Error

4.12.2.92 #define ENET_RINFO_MCAST 0x00200000

Multicast Frame

4.12.2.93 #define ENET_RINFO_NO_DESCR 0x20000000

No new Descriptor available

4.12.2.94 #define ENET_RINFO_OVERRUN 0x10000000

Receive overrun

4.12.2.95 #define ENET_RINFO_RANGE_ERR 0x04000000

Range Error (exceeded max. size)

4.12.2.96 #define ENET_RINFO_SIZE(*n*) (((n) & 0x7FF) + 1)

Data size in bytes

4.12.2.97 #define ENET_RINFO_SYM_ERR 0x01000000

Symbol Error from PHY

4.12.2.98 #define ENET_RINFO_VLAN 0x00080000

VLAN Frame

4.12.2.99 #define ENET_RSV_BROADCAST 0x02000000

Broadcast Frame

4.12.2.100 #define ENET_RSV_CESEEN 0x00040000

Carrier Event Previously Seen

4.12.2.101 #define ENET_RSV_CONTROLFRAME 0x08000000

Control Frame

4.12.2.102 #define ENET_RSV_CRCERR 0x00100000

CRC Error

4.12.2.103 #define ENET_RSV_DRIBBLENIBBLE 0x04000000

Dribble Nibble

4.12.2.104 #define ENET_RSV_LCERR 0x00200000

Length Check Error

4.12.2.105 #define ENET_RSV_LOR 0x00400000

Length Out of Range

4.12.2.106 #define ENET_RSV_MULTICAST 0x01000000

Multicast Frame

4.12.2.107 #define ENET_RSV_PAUSE 0x10000000

Pause Frame

4.12.2.108 #define ENET_RSV_PPI 0x00010000

Packet Previously Ignored

4.12.2.109 #define ENET_RSV_RBC 0x0000FFFF

Receive Byte Count

4.12.2.110 #define ENET_RSV_RCV 0x00080000

Receive Code Violation

4.12.2.111 #define ENET_RSV_ROK 0x00800000

Frame Received OK

4.12.2.112 #define ENET_RSV_RXDVSEEN 0x00020000

RXDV Event Previously Seen

4.12.2.113 #define ENET_RSV_UO 0x20000000

Unsupported Opcode

4.12.2.114 #define ENET_RSV_VLAN 0x40000000

VLAN Frame

4.12.2.115 #define ENET_RXFILTERCTRL_ABE 0x00000002

Accept Broadcast Frames Enable

4.12.2.116 #define ENET_RXFILTERCTRL_AME 0x00000004

Accept Multicast Frames Enable

4.12.2.117 #define ENET_RXFILTERCTRL_AMHE 0x00000010

Accept Multicast Hash Filter Fram

4.12.2.118 #define ENET_RXFILTERCTRL_APE 0x00000020

Accept Perfect Match Enable

4.12.2.119 #define ENET_RXFILTERCTRL_AUE 0x00000001

Accept Unicast Frames Enable

4.12.2.120 #define ENET_RXFILTERCTRL_AUHE 0x00000008

Accept Unicast Hash Filter Frames

4.12.2.121 #define ENET_RXFILTERCTRL_MPEW 0x00001000

Magic Packet Filter WoL Enable

4.12.2.122 #define ENET_RXFILTERCTRL_RFEW 0x00002000

Perfect Filter WoL Enable

4.12.2.123 #define ENET_RXFILTERWOLSTATUS_ABW 0x00000002

Broadcast Frame caused WoL

4.12.2.124 #define ENET_RXFILTERWOLSTATUS_AMHW 0x00000010

Multicast Hash Filter Frame WoL

4.12.2.125 #define ENET_RXFILTERWOLSTATUS_AMW 0x00000004

Multicast Frame caused WoL

4.12.2.126 #define ENET_RXFILTERWOLSTATUS_APW 0x00000020

Perfect Filter WoL

4.12.2.127 #define ENET_RXFILTERWOLSTATUS_AUHW 0x00000008

Unicast Hash Filter Frame WoL

4.12.2.128 #define ENET_RXFILTERWOLSTATUS_AUW 0x00000001

Unicast Frame caused WoL

4.12.2.129 #define ENET_RXFILTERWOLSTATUS_BITMASK 0x01BF

Receive Filter WoL Status/Clear bitmasl value

4.12.2.130 #define ENET_RXFILTERWOLSTATUS_MPW 0x00000100

Magic Packet Filter caused WoL

4.12.2.131 #define ENET_RXFILTERWOLSTATUS_RFW 0x00000080

RX Filter caused WoL

4.12.2.132 #define ENET_STATUS_RXSTATUS 0x00000001

Receive Channel Active Status

4.12.2.133 #define ENET_STATUS_TXSTATUS 0x00000002

Transmit Channel Active Status

4.12.2.134 #define ENET_SUPP_100Mbps_SPEED 0x00000100

Reduced MII Logic Current Speed

4.12.2.135 #define ENET_TCTRL_CRC 0x20000000

Append a hardware CRC to Frame

4.12.2.136 #define ENET_TCTRL_HUGE 0x08000000

Enable Huge Frame

4.12.2.137 #define ENET_TCTRL_INT 0x80000000

Generate TxDone Interrupt

4.12.2.138 #define ENET_TCTRL_LAST 0x40000000

Last Descriptor for TX Frame

4.12.2.139 #define ENET_TCTRL_OVERRIDE 0x04000000

Override Default MAC Registers

4.12.2.140 #define ENET_TCTRL_PAD 0x10000000

Pad short Frames to 64 bytes

4.12.2.141 #define ENET_TCTRL_SIZE(*n*) (((n) - 1) & 0x7FF)

Size of data buffer in bytes

4.12.2.142 #define ENET_TEST_SCPQ 0x00000001

Shortcut Pause Quanta

4.12.2.143 #define ENET_TEST_TESTBP 0x00000004

Test Back Pressure

4.12.2.144 #define ENET_TEST_TESTPAUSE 0x00000002

Test Pause

4.12.2.145 #define ENET_TINFO_COL_CNT 0x01E00000

Collision Count

4.12.2.146 #define ENET_TINFO_DEFER 0x02000000

Packet Deferred (not an error)

4.12.2.147 #define ENET_TINFO_ERR 0x80000000

Error Occured (OR of all errors)

4.12.2.148 #define ENET_TINFO_EXCESS_COL 0x08000000

Excessive Collision

4.12.2.149 #define ENET_TINFO_EXCESS_DEF 0x04000000

Excessive Deferral

4.12.2.150 #define ENET_TINFO_LATE_COL 0x10000000

Late Collision Occured

4.12.2.151 #define ENET_TINFO_NO_DESCR 0x40000000

No new Descriptor available

4.12.2.152 #define ENET_TINFO_UNDERRUN 0x20000000

Transmit Underrun

4.12.2.153 #define ENET_TSV0_BACKPRESSURE 0x40000000

Backpressure Method Applied

4.12.2.154 #define ENET_TSV0_BROADCAST 0x00000020

Broadcast Destination

4.12.2.155 #define ENET_TSV0_CONTROLFRAME 0x10000000

Control Frame

4.12.2.156 #define ENET_TSV0_CRCERR 0x00000001

CRC error

4.12.2.157 #define ENET_TSV0_DONE 0x00000008

Tramsmission Completed

4.12.2.158 #define ENET_TSV0_EXCOL 0x00000100

Excessive Collision

4.12.2.159 #define ENET_TSV0_EXDF 0x00000080

Excessive Packet Deferral

4.12.2.160 #define ENET_TSV0_GIANT 0x00000400

Giant Frame

4.12.2.161 #define ENET_TSV0_LCE 0x00000002

Length Check Error

4.12.2.162 #define ENET_TSV0_LCOL 0x00000200

Late Collision Occured

4.12.2.163 #define ENET_TSV0_LOR 0x00000004

Length Out of Range

4.12.2.164 #define ENET_TSV0_MULTICAST 0x00000010

Multicast Destination

4.12.2.165 #define ENET_TSV0_PACKETDEFER 0x00000040

Packet Deferred

4.12.2.166 #define ENET_TSV0_PAUSE 0x20000000

Pause Frame

4.12.2.167 #define ENET_TSV0_TOTALBYTES 0x0FFFF000

Total Bytes Transferred

4.12.2.168 #define ENET_TSV0_UNDERRUN 0x00000800

Buffer Underrun

4.12.2.169 #define ENET_TSV0_VLAN 0x80000000

VLAN Frame

4.12.2.170 #define ENET_TSV1_TBC 0x0000FFFF

Transmit Byte Count

4.12.2.171 #define ENET_TSV1_TCC 0x000F0000

Transmit Collision Count

4.12.3 Function Documentation

4.12.3.1 STATIC INLINE void Chip_ENET_ClearIntStatus (LPC_ENET_T * *pENET,* uint32_t *mask*)

Clear the interrupt status.

Parameters

pENET	: The base of ENET peripheral on the chip
mask	: Interrupt mask (Or-ed bit values of ENET_INT_*)

Returns

Nothing

4.12.3.2 void Chip_ENET_DeInit (LPC_ENET_T * *pENET*)

De-initialize the ethernet interface.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

4.12.3.3 STATIC INLINE void Chip_ENET_DisableInt (LPC_ENET_T * *pENET,* uint32_t *mask*)

Disable ENET interrupts.

Parameters

pENET	: The base of ENET peripheral on the chip
mask	: Interrupt mask (Or-ed bit values of ENET_INT_*)

Returns

Nothing

4.12.3.4 STATIC INLINE void Chip_ENET_DisableRXFilter (LPC_ENET_T * *pENET,* uint32_t *mask*)

Disable Rx Filter.

Parameters

pENET	: The base of ENET peripheral on the chip
mask	: Filter mask (Or-ed bit values of ENET_RXFILTERCTRL_∗)

Returns

Nothing

4.12.3.5 STATIC INLINE void Chip_ENET_EnableInt (LPC_ENET_T ∗ *pENET*, uint32_t *mask*)

Enable ENET interrupts.

Parameters

pENET	: The base of ENET peripheral on the chip
mask	: Interrupt mask (Or-ed bit values of ENET_INT_∗)

Returns

Nothing

4.12.3.6 STATIC INLINE void Chip_ENET_EnableRXFilter (LPC_ENET_T ∗ *pENET*, uint32_t *mask*)

Enable Rx Filter.

Parameters

pENET	: The base of ENET peripheral on the chip
mask	: Filter mask (Or-ed bit values of ENET_RXFILTERCTRL_∗)

Returns

Nothing

4.12.3.7 uint32_t Chip_ENET_FindMIIDiv (LPC_ENET_T ∗ *pENET*, uint32_t *clockRate*)

Find the divider index for a desired MII clock rate.

Parameters

pENET	: The base of ENET peripheral on the chip
clockRate	: Clock rate to get divider index for

Returns

MII divider index to get the closest clock rate for clockRate

Note

Use this function to get a divider index for the Chip_ENET_SetupMII() function determined from the desired MII clock rate.

4.12.3.8 ENET_BUFF_STATUS_T Chip_ENET_GetBufferStatus (LPC_ENET_T ∗ *pENET*, uint16_t *produceIndex*, uint16_t *consumeIndex*, uint16_t *buffSize*)

Get the buffer status with the current Produce Index and Consume Index.

Parameters

pENET	: The base of ENET peripheral on the chip
produceIndex	: Produce Index
consumeIndex	: Consume Index
buffSize	: Buffer size

Returns

Status (One of status value: ENET_BUFF_EMPTY/ENET_BUFF_FULL/ENET_BUFF_PARTIAL_FULL)

4.12.3.9 uint32_t Chip_ENET_GetFillDescNum (LPC_ENET_T * *pENET*, uint16_t *produceIndex*, uint16_t *consumeIndex*, uint16_t *buffSize*)

Get the number of descriptors filled.

Parameters

pENET	: The base of ENET peripheral on the chip
produceIndex	: Produce Index
consumeIndex	: Consume Index
buffSize	: Buffer size

Returns

the number of descriptors

4.12.3.10 STATIC INLINE uint32_t Chip_ENET_GetFreeDescNum (LPC_ENET_T * *pENET*, uint16_t *produceIndex*, uint16_t *consumeIndex*, uint16_t *buffSize*)

Get the number of free descriptors.

Parameters

pENET	: The base of ENET peripheral on the chip
produceIndex	: Produce Index
consumeIndex	: Consume Index
buffSize	: Buffer size

Returns

the number of descriptors

4.12.3.11 STATIC INLINE uint32_t Chip_ENET_GetIntStatus (LPC_ENET_T * *pENET*)

Get the interrupt status.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

interrupt status (Or-ed bit values of ENET_INT_*)

4.12.3.12 STATIC INLINE uint16_t Chip_ENET_GetRXConsumeIndex (LPC_ENET_T * *pENET*)

Get the current Rx Consume Descriptor Index.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Rx Consume Index

4.12.3.13 STATIC INLINE uint16_t Chip_ENET_GetRXProduceIndex (LPC_ENET_T * *pENET*)

Get the current Rx Produce Descriptor Index.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Rx Produce Index

4.12.3.14 STATIC INLINE uint16_t Chip_ENET_GetTXConsumeIndex (LPC_ENET_T * *pENET*)

Get the current Tx Consume Descriptor Index.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Tx Consume Index

4.12.3.15 STATIC INLINE uint16_t Chip_ENET_GetTXProduceIndex (LPC_ENET_T * *pENET*)

Get the current Tx Produce Descriptor Index.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Tx Produce Index

4.12.3.16 uint16_t Chip_ENET_IncRXConsumeIndex (LPC_ENET_T * *pENET*)

Increase the current Rx Consume Descriptor Index.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

The new index value

4.12.3.17 uint16_t Chip_ENET_IncTXProduceIndex (LPC_ENET_T * *pENET*)

Increase the current Tx Produce Descriptor Index.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

The new index value

4.12.3.18 void Chip_ENET_Init (LPC_ENET_T * *pENET*, bool *useRMII*)

Initialize ethernet interface.

Parameters

pENET	: The base of ENET peripheral on the chip
useRMII	: true to setup interface for RMII, false for MII

Returns

Nothing

Note

Performs basic initialization of the ethernet interface in a default state. This is enough to place the interface in a usable state, but may require more setup outside this function.

4.12.3.19 void Chip_ENET_InitRxDescriptors (LPC_ENET_T * *pENET*, ENET_RXDESC_T * *pDescs*, ENET_RXSTAT_T * *pStatus*, uint32_t *descNum*)

Configures the initial ethernet receive descriptors.

Parameters

pENET	: The base of ENET peripheral on the chip
pDescs	: Pointer to TX descriptor list
pStatus	: Pointer to TX status list
descNum	: the number of desciptors

Returns

Nothing

4.12.3.20 void Chip_ENET_InitTxDescriptors (LPC_ENET_T * *pENET*, ENET_TXDESC_T * *pDescs*, ENET_TXSTAT_T * *pStatus*, uint32_t *descNum*)

Configures the initial ethernet transmit descriptors.

Parameters

pENET	: The base of ENET peripheral on the chip
pDescs	: Pointer to TX descriptor list
pStatus	: Pointer to TX status list
descNum	: the number of desciptors

Returns

Nothing

4.12.3.21 STATIC INLINE bool Chip_ENET_IsMIIBusy (LPC_ENET_T * *pENET*)

Returns MII link (PHY) busy status.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Returns true if busy, otherwise false

4.12.3.22 STATIC INLINE bool Chip_ENET_IsRxEmpty (LPC_ENET_T * *pENET*)

Check if Rx buffer is empty.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

true/false

4.12.3.23 STATIC INLINE bool Chip_ENET_IsTxFull (LPC_ENET_T * *pENET*)

Check if Tx buffer is full.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

true/false

4.12.3.24 uint16_t Chip_ENET_ReadMIIData (LPC_ENET_T * *pENET*)

Returns the value read from the PHY.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Read value from PHY

4.12.3.25 STATIC INLINE void Chip_ENET_Reset (LPC_ENET_T * *pENET*)

Resets the ethernet interface.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

Note

Resets the ethernet interface. This should be called prior to Chip_ENET_Init with a small delay after this call.

4.12.3.26 STATIC INLINE void Chip_ENET_ResetRXLogic (LPC_ENET_T * *pENET*)

Reset Rx Logic.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

4.12.3.27 STATIC INLINE void Chip_ENET_ResetTXLogic (LPC_ENET_T * *pENET*)

Reset Tx Logic.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

4.12.3.28 STATIC INLINE void Chip_ENET_RXDisable (LPC_ENET_T * *pENET*)

Disables ethernet packet reception.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

4.12.3.29 STATIC INLINE void Chip_ENET_RXEnable (LPC_ENET_T * *pENET*)

Enables ethernet packet reception.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

4.12.3.30 STATIC INLINE void Chip_ENET_Set100Mbps (LPC_ENET_T * *pENET*)

Selects 100Mbps for the current speed.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

4.12.3.31 STATIC INLINE void Chip_ENET_Set10Mbps (LPC_ENET_T * *pENET*)

Selects 10Mbps for the current speed.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

4.12.3.32 STATIC INLINE void Chip_ENET_SetADDR (LPC_ENET_T * *pENET,* const uint8_t * *macAddr*)

Sets the address of the interface.

Parameters

pENET	: The base of ENET peripheral on the chip

macAddr	: Pointer to the 6 bytes used for the MAC address

Returns

Nothing

4.12.3.33 void Chip_ENET_SetFullDuplex (LPC_ENET_T * *pENET*)

Sets full duplex operation for the interface.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

4.12.3.34 void Chip_ENET_SetHalfDuplex (LPC_ENET_T * *pENET*)

Sets half duplex operation for the interface.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

4.12.3.35 void Chip_ENET_SetupMII (LPC_ENET_T * *pENET,* uint32_t *div,* uint8_t *addr*)

Sets up the PHY link clock divider and PHY address.

Parameters

pENET	: The base of ENET peripheral on the chip
div	: Divider index, not a divider value, see user manual
addr	: PHY address, used with MII read and write

Returns

Nothing

Note

The MII clock divider rate is divided from the peripheral clock returned from the Chip_Clock_GetSystem↩ ClockRate() function. Use Chip_ENET_FindMIIDiv() with a desired clock rate to find the correct divider index value.

4.12.3.36 void Chip_ENET_StartMIIRead (LPC_ENET_T * *pENET,* uint8_t *reg*)

Starts a PHY read via the MII.

Parameters

pENET	: The base of ENET peripheral on the chip
reg	: PHY register to read

Returns

Nothing

Note

Start a PHY read operation. Does not block, requires calling IP_ENET_IsMIIBusy to determine when read is complete and calling IP_ENET_ReadMIIData to get the data.

4.12.3.37 void Chip_ENET_StartMIIWrite (LPC_ENET_T * *pENET*, uint8_t *reg*, uint16_t *data*)

Starts a PHY write via the MII.

Parameters

pENET	: The base of ENET peripheral on the chip
reg	: PHY register to write
data	: Data to write to PHY register

Returns

Nothing

Note

Start a PHY write operation. Does not block, requires calling IP_ENET_IsMIIBusy to determine when write is complete.

4.12.3.38 STATIC INLINE void Chip_ENET_TXDisable (LPC_ENET_T * *pENET*)

Disables ethernet transmit.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

4.12.3.39 STATIC INLINE void Chip_ENET_TXEnable (LPC_ENET_T * *pENET*)

Enables ethernet transmit.

Parameters

pENET	: The base of ENET peripheral on the chip

Returns

Nothing

4.13 : Common GPIO functions

Functions

- unsigned int GPIO_Read (void)

  Read from GPIO ports.
- void GPIO_Init (unsigned int pinMap)

  Enable GPIO ports.
- void GPIO_SetDir (unsigned int pinMap, unsigned int pinDirectionMap)

  Initialize directions for GPIO ports, input or output.
- void GPIO_Write (unsigned int preparedValue, unsigned int pinMap)

  Write on GPIO ports.

4.13.1 Detailed Description

This file contains common GPIO functions.

4.13.2 Function Documentation

4.13.2.1 void GPIO_Init (unsigned int *pinMap*)

Enable GPIO ports.

Parameters

pinMap	to enable GPIO ports

Returns

Nothing

4.13.2.2 unsigned int GPIO_Read (void)

Read from GPIO ports.

Parameters

Nothing	

Returns

Value in GPIO ports

4.13.2.3 void GPIO_SetDir (unsigned int *pinMap*, unsigned int *pinDirectionMap*)

Initialize directions for GPIO ports, input or output.

Parameters

GPIO	ports to select

directions	for selected GPIO ports

Returns

Nothing

4.13.2.4 void GPIO_Write (unsigned int *preparedValue,* unsigned int *pinMap*)

Write on GPIO ports.

Parameters

value	to be written on selected ports
ports	to be selected for write operation

Returns

Nothing

4.14 : Common I2C functions

Functions

- void I2C_Init (void)

  Make initialization for I2C communication.
- unsigned int I2C_Transfer (unsigned char addr, int read, void *data, unsigned int size, int freq)

  Transfer data via I2C communication.

4.14.1 Detailed Description

This file contains common I2C functions.

4.14.2 Function Documentation

4.14.2.1 void I2C_Init (void)

Make initialization for I2C communication.

Parameters

Nothing	

Returns

Nothing

4.14.2.2 unsigned int I2C_Transfer (unsigned char *addr*, int *read*, void * *data*, unsigned int *size*, int *freq*)

Transfer data via I2C communication.

Parameters

address	
direction->	read =1 or write=0
data	to be written
data	size
communication's	frequency

Returns

Response from an I2C transaction

4.15 CHIP: Common Chip ISP/IAP commands and return codes

Macros

- #define IAP_PREWRRITE_CMD 50
- #define IAP_WRISECTOR_CMD 51
- #define IAP_ERSSECTOR_CMD 52
- #define IAP_BLANK_CHECK_SECTOR_CMD 53
- #define IAP_REPID_CMD 54
- #define IAP_READ_BOOT_CODE_CMD 55
- #define IAP_COMPARE_CMD 56
- #define IAP_REINVOKE_ISP_CMD 57
- #define IAP_READ_UID_CMD 58
- #define IAP_ERASE_PAGE_CMD 59
- #define IAP_EEPROM_WRITE 61
- #define IAP_EEPROM_READ 62
- #define IAP_CMD_SUCCESS 0
- #define IAP_INVALID_COMMAND 1
- #define IAP_SRC_ADDR_ERROR 2
- #define IAP_DST_ADDR_ERROR 3
- #define IAP_SRC_ADDR_NOT_MAPPED 4
- #define IAP_DST_ADDR_NOT_MAPPED 5
- #define IAP_COUNT_ERROR 6
- #define IAP_INVALID_SECTOR 7
- #define IAP_SECTOR_NOT_BLANK 8
- #define IAP_SECTOR_NOT_PREPARED 9
- #define IAP_COMPARE_ERROR 10
- #define IAP_BUSY 11
- #define IAP_PARAM_ERROR 12
- #define IAP_ADDR_ERROR 13
- #define IAP_ADDR_NOT_MAPPED 14
- #define IAP_CMD_LOCKED 15
- #define IAP_INVALID_CODE 16
- #define IAP_INVALID_BAUD_RATE 17
- #define IAP_INVALID_STOP_BIT 18
- #define IAP_CRP_ENABLED 19

Typedefs

- typedef void(∗ **IAP_ENTRY_T**)(unsigned int[5], unsigned int[4])

Functions

- uint8_t Chip_IAP_PreSectorForReadWrite (uint32_t strSector, uint32_t endSector)

  Prepare sector for write operation.
- uint8_t Chip_IAP_CopyRamToFlash (uint32_t dstAdd, uint32_t ∗srcAdd, uint32_t byteswrt)

  Copy RAM to flash.
- uint8_t Chip_IAP_EraseSector (uint32_t strSector, uint32_t endSector)

  Erase sector.
- uint8_t Chip_IAP_BlankCheckSector (uint32_t strSector, uint32_t endSector)

  Blank check a sector or multiples sector of on-chip flash memory.
- uint32_t Chip_IAP_ReadPID (void)

  Read part identification number.

- uint8_t Chip_IAP_ReadBootCode (void)

  Read boot code version number.
- uint8_t Chip_IAP_Compare (uint32_t dstAdd, uint32_t srcAdd, uint32_t bytescmp)

  Compare the memory contents at two locations.
- uint8_t Chip_IAP_ReinvokeISP (void)

  IAP reinvoke ISP to invoke the bootloader in ISP mode.
- uint32_t Chip_IAP_ReadUID (void)

  Read the unique ID.
- uint8_t Chip_IAP_ErasePage (uint32_t strPage, uint32_t endPage)

  Erase a page or multiple papers of on-chip flash memory.

4.15.1 Detailed Description

4.15.2 Macro Definition Documentation

4.15.2.1 #define IAP_ADDR_ERROR 13

Address is not on word boundary

4.15.2.2 #define IAP_ADDR_NOT_MAPPED 14

Address is not mapped in the memory map

4.15.2.3 #define IAP_BLANK_CHECK_SECTOR_CMD 53

Blank check sector

4.15.2.4 #define IAP_BUSY 11

Flash programming hardware interface is busy

4.15.2.5 #define IAP_CMD_LOCKED 15

Command is locked

4.15.2.6 #define IAP_CMD_SUCCESS 0

Command is executed successfully

4.15.2.7 #define IAP_COMPARE_CMD 56

Compare two RAM address locations

4.15.2.8 #define IAP_COMPARE_ERROR 10

Source and destination data not equal

4.15.2.9 #define IAP_COUNT_ERROR 6

Byte count is not multiple of 4 or is not a permitted value

4.15.2.10 #define IAP_CRP_ENABLED 19

Code read protection enabled

4.15.2.11 #define IAP_DST_ADDR_ERROR 3

Destination address is not on a correct boundary

4.15.2.12 #define IAP_DST_ADDR_NOT_MAPPED 5

Destination address is not mapped in the memory map

4.15.2.13 #define IAP_EEPROM_READ 62

EEPROM READ command

4.15.2.14 #define IAP_EEPROM_WRITE 61

EEPROM Write command

4.15.2.15 #define IAP_ERASE_PAGE_CMD 59

Erase page

4.15.2.16 #define IAP_ERSSECTOR_CMD 52

Erase Sector command

4.15.2.17 #define IAP_INVALID_BAUD_RATE 17

Invalid baud rate setting

4.15.2.18 #define IAP_INVALID_CODE 16

Unlock code is invalid

4.15.2.19 #define IAP_INVALID_COMMAND 1

Invalid command

4.15.2.20 #define IAP_INVALID_SECTOR 7

Sector number is invalid or end sector number is greater than start sector number

4.15.2.21 #define IAP_INVALID_STOP_BIT 18

Invalid stop bit setting

4.15.2.22 #define IAP_PARAM_ERROR 12

nsufficient number of parameters or invalid parameter

4.15.2.23 #define IAP_PREWRRITE_CMD 50

Prepare sector for write operation command

4.15.2.24 #define IAP_READ_BOOT_CODE_CMD 55

Read Boot code version

4.15.2.25 #define IAP_READ_UID_CMD 58

Read UID

4.15.2.26 #define IAP_REINVOKE_ISP_CMD 57

Reinvoke ISP

4.15.2.27 #define IAP_REPID_CMD 54

Read PartID command

4.15.2.28 #define IAP_SECTOR_NOT_BLANK 8

Sector is not blank

4.15.2.29 #define IAP_SECTOR_NOT_PREPARED 9

Command to prepare sector for write operation was not executed

4.15.2.30 #define IAP_SRC_ADDR_ERROR 2

Source address is not on word boundary

4.15.2.31 #define IAP_SRC_ADDR_NOT_MAPPED 4

Source address is not mapped in the memory map

4.15.2.32 #define IAP_WRISECTOR_CMD 51

Write Sector command

4.15.3 Function Documentation

4.15.3.1 uint8_t Chip_IAP_BlankCheckSector (uint32_t *strSector,* uint32_t *endSector*)

Blank check a sector or multiples sector of on-chip flash memory.

Parameters

strSector	: Start sector number
endSector	: End sector number

Returns

Offset of the first non blank word location if the status code is SECTOR_NOT_BLANK

Note

The end sector must be greater than or equal to start sector number

4.15.3.2 uint8_t Chip_IAP_Compare (uint32_t *dstAdd,* uint32_t *srcAdd,* uint32_t *bytescmp*)

Compare the memory contents at two locations.

Parameters

dstAdd	: Destination of the RAM address of data bytes to be compared
srcAdd	: Source of the RAM address of data bytes to be compared
bytescmp	: Number of bytes to be compared

Returns

Offset of the first mismatch of the status code is COMPARE_ERROR

Note

The addresses should be a word boundary and number of bytes should be a multiply of 4

4.15.3.3 uint8_t Chip_IAP_CopyRamToFlash (uint32_t *dstAdd,* uint32_t * *srcAdd,* uint32_t *byteswrt*)

Copy RAM to flash.

Parameters

dstAdd	: Destination flash address where data bytes are to be written
srcAdd	: Source flash address where data bytes are to be read
byteswrt	: Number of bytes to be written

Returns

Status code to indicate the command is executed successfully or not

Note

The addresses should be a 256 byte boundary and the number of bytes should be 256 | 512 | 1024 | 4096

4.15.3.4 uint8_t Chip_IAP_ErasePage (uint32_t *strPage,* uint32_t *endPage*)

Erase a page or multiple papers of on-chip flash memory.

Parameters

strPage	: Start page number
endPage	: End page number

Returns

Status code to indicate the command is executed successfully or not

Note

The page number must be greater than or equal to start page number

4.15.3.5 uint8_t Chip_IAP_EraseSector (uint32_t *strSector,* uint32_t *endSector*)

Erase sector.

Parameters

strSector	: Start sector number
endSector	: End sector number

Returns

Status code to indicate the command is executed successfully or not

Note

The end sector must be greater than or equal to start sector number

4.15.3.6 uint8_t Chip_IAP_PreSectorForReadWrite (uint32_t *strSector,* uint32_t *endSector*)

Prepare sector for write operation.

Parameters

strSector	: Start sector number
endSector	: End sector number

Returns

Status code to indicate the command is executed successfully or not

Note

This command must be executed before executing "Copy RAM to flash" or "Erase Sector" command. The end sector must be greater than or equal to start sector number

4.15.3.7 uint8_t Chip_IAP_ReadBootCode (void)

Read boot code version number.

Returns

Boot code version number

4.15.3.8 uint32_t Chip_IAP_ReadPID (void)

Read part identification number.

Returns

Part identification number

4.15.3.9 uint32_t Chip_IAP_ReadUID (void)

Read the unique ID.

Returns

Status code to indicate the command is executed successfully or not

4.15.3.10 uint8_t Chip_IAP_ReinvokeISP (void)

IAP reinvoke ISP to invoke the bootloader in ISP mode.

Returns

none

4.16 CHIP: LPC17xx/40xx I/O configuration driver

Classes

- struct PINMUX_GRP_T

  Array of IOCON pin definitions passed to Chip_IOCON_SetPinMuxing() must be in this format.
- struct LPC_IOCON_T

  IOCON register block.

Macros

- #define IOCON_FUNC0 0x0
- #define IOCON_FUNC1 0x1
- #define IOCON_FUNC2 0x2
- #define IOCON_FUNC3 0x3
- #define IOCON_FUNC4 0x4
- #define IOCON_FUNC5 0x5
- #define IOCON_FUNC6 0x6
- #define IOCON_FUNC7 0x7
- #define IOCON_MODE_INACT (0x0 << 3)
- #define IOCON_MODE_PULLDOWN (0x1 << 3)
- #define IOCON_MODE_PULLUP (0x2 << 3)
- #define IOCON_MODE_REPEATER (0x3 << 3)
- #define IOCON_HYS_EN (0x1 << 5)
- #define IOCON_INV_EN (0x1 << 6)
- #define IOCON_ADMODE_EN (0x0 << 7)
- #define IOCON_DIGMODE_EN (0x1 << 7)
- #define IOCON_FILT_DIS (0x1 << 8)
- #define IOCON_HS_DIS (0x1 << 8)
- #define IOCON_HIDRIVE_EN (0x1 << 9)
- #define IOCON_FASTSLEW_EN (0x1 << 9)
- #define IOCON_OPENDRAIN_EN (0x1 << 10)
- #define IOCON_DAC_EN (0x1 << 16)
- #define FUNC0 0x0 /** Function 0 */
- #define **FUNC1** 0x1 /** Function 1 */
- #define **FUNC2** 0x2 /** Function 2 */
- #define **FUNC3** 0x3 /** Function 3 */
- #define **MD_PLN** (0x0 << 3)
- #define **MD_PDN** (0x1 << 3)
- #define **MD_PUP** (0x2 << 3)
- #define **MD_BUK** (0x3 << 3)
- #define MD_HYS_ENA (0x1 << 5)
- #define MD_HYS_DIS (0x0 << 5)
- #define MD_IINV_ENA (0x1 << 6)
- #define MD_IINV_DIS (0x0 << 6)
- #define MD_OD_ENA (0x1 << 10)
- #define MD_OD_DIS (0x0 << 10)
- #define MD_HS_ENA (0x0 << 8)
- #define MD_HS_DIS (0x1 << 8)
- #define MD_ANA_ENA (0x0 << 7)
- #define MD_ANA_DIS (0x1 << 7)
- #define MD_FILT_ENA (0x0 << 8)
- #define MD_FILT_DIS (0x1 << 8)
- #define MD_DAC_ENA (0x1 << 16)

- #define MD_DAC_DIS (0x0 << 16)
- #define MD_STD_SLEW_RATE (0x0 << 9)
- #define MD_FAST_SLEW_RATE (0x1 << 9)
- #define MD_HD_ENA (0x1 << 9)
- #define MD_HD_DIS (0x0 << 9)
- #define **FUNC4** 0x4 /** Function 4 */
- #define **FUNC5** 0x5 /** Function 5 */
- #define **FUNC6** 0x6 /** Function 6 */
- #define **FUNC7** 0x7 /** Function 7 */

Functions

- STATIC INLINE void Chip_IOCON_Init (LPC_IOCON_T *pIOCON)

  Initialize the IOCON peripheral.
- STATIC INLINE void Chip_IOCON_PinMuxSet (LPC_IOCON_T *pIOCON, uint8_t port, uint8_t pin, uint32_t modefunc)

  Sets I/O Control pin mux.
- STATIC INLINE void Chip_IOCON_PinMux (LPC_IOCON_T *pIOCON, uint8_t port, uint8_t pin, uint32_t mode, uint8_t func)

  Setup pin modes and function.
- void Chip_IOCON_SetPinMuxing (LPC_IOCON_T *pIOCON, const PINMUX_GRP_T *pinArray, uint32_t arrayLength)

  Set all I/O Control pin muxing.

4.16.1 Detailed Description

4.16.2 Macro Definition Documentation

4.16.2.1 #define FUNC0 0x0 /** Function 0 */

IOCON function and mode selection definitions (old) For backwards compatibility.

4.16.2.2 #define IOCON_ADMODE_EN (0x0 << 7)

Enables analog input function (analog pins only)

4.16.2.3 #define IOCON_DAC_EN (0x1 << 16)

Enables DAC function

4.16.2.4 #define IOCON_DIGMODE_EN (0x1 << 7)

Enables digital function (analog pins only)

4.16.2.5 #define IOCON_FASTSLEW_EN (0x1 << 9)

Enables fast slew

4.16.2.6 #define IOCON_FILT_DIS (0x1 << 8)

Disables noise pulses filtering (10nS glitch filter)

4.16.2.7 #define IOCON_FUNC0 0x0

IOCON function and mode selection definitions See the User Manual for specific modes and functions supoprted by the various LPC11xx devices. Functionality can vary per device.Selects pin function 0

4.16.2.8 #define IOCON_FUNC1 0x1

Selects pin function 1

4.16.2.9 #define IOCON_FUNC2 0x2

Selects pin function 2

4.16.2.10 #define IOCON_FUNC3 0x3

Selects pin function 3

4.16.2.11 #define IOCON_FUNC4 0x4

Selects pin function 4

4.16.2.12 #define IOCON_FUNC5 0x5

Selects pin function 5

4.16.2.13 #define IOCON_FUNC6 0x6

Selects pin function 6

4.16.2.14 #define IOCON_FUNC7 0x7

Selects pin function 7

4.16.2.15 #define IOCON_HIDRIVE_EN (0x1 << 9)

Sink current is 20 mA

4.16.2.16 #define IOCON_HS_DIS (0x1 << 8)

I2C glitch filter and slew rate disabled

4.16.2.17 #define IOCON_HYS_EN (0x1 << 5)

Enables hysteresis

4.16.2.18 #define IOCON_INV_EN (0x1 << 6)

Enables invert function on input

4.16.2.19 #define IOCON_MODE_INACT (0x0 << 3)

No addition pin function

4.16.2.20 #define IOCON_MODE_PULLDOWN (0x1 << 3)

Selects pull-down function

4.16.2.21 #define IOCON_MODE_PULLUP (0x2 << 3)

Selects pull-up function

4.16.2.22 #define IOCON_MODE_REPEATER (0x3 << 3)

Selects pin repeater function

4.16.2.23 #define IOCON_OPENDRAIN_EN (0x1 << 10)

Enables open-drain function

4.16.2.24 #define MD_ANA_DIS (0x1 << 7)

Macro to disable analog mode (ADC)- use with Chip_IOCON_PinMux

4.16.2.25 #define MD_ANA_ENA (0x0 << 7)

Macro to enable analog mode (ADC)- use with Chip_IOCON_PinMux

4.16.2.26 #define MD_DAC_DIS (0x0 << 16)

Macro to disable DAC- use with Chip_IOCON_PinMux

4.16.2.27 #define MD_DAC_ENA (0x1 << 16)

Macro to enable DAC- use with Chip_IOCON_PinMux

4.16.2.28 #define MD_FAST_SLEW_RATE (0x1 << 9)

Macro to enable fast mode, slew rate control is disabled - use with Chip_IOCON_PinMux

4.16.2.29 #define MD_FILT_DIS (0x1 << 8)

Macro to disable input filter- use with Chip_IOCON_PinMux

4.16.2.30 #define MD_FILT_ENA (0x0 << 8)

Macro to enable input filter- use with Chip_IOCON_PinMux

4.16.2.31 #define MD_HD_DIS (0x0 << 9)

Macro to disable high drive output- use with Chip_IOCON_PinMux

4.16.2.32 #define MD_HD_ENA (0x1 << 9)

Macro to enable high drive output- use with Chip_IOCON_PinMux

4.16.2.33 #define MD_HS_DIS (0x1 << 8)

Macro to disable I2C 50ns glitch filter and slew rate control- use with Chip_IOCON_PinMux

4.16.2.34 #define MD_HS_ENA (0x0 << 8)

Macro to enable I2C 50ns glitch filter and slew rate control- use with Chip_IOCON_PinMux

4.16.2.35 #define MD_HYS_DIS (0x0 << 5)

Macro to disable hysteresis- use with Chip_IOCON_PinMux

4.16.2.36 #define MD_HYS_ENA (0x1 << 5)

Macro to enable hysteresis- use with Chip_IOCON_PinMux

4.16.2.37 #define MD_IINV_DIS (0x0 << 6)

Macro to disable input inversion- use with Chip_IOCON_PinMux

4.16.2.38 #define MD_IINV_ENA (0x1 << 6)

Macro to enable input inversion- use with Chip_IOCON_PinMux

4.16.2.39 #define MD_OD_DIS (0x0 << 10)

Macro to disable simulated open drain mode- use with Chip_IOCON_PinMux

4.16.2.40 #define MD_OD_ENA (0x1 << 10)

Macro to enable simulated open drain mode- use with Chip_IOCON_PinMux

4.16.2.41 #define MD_STD_SLEW_RATE (0x0 << 9)

Macro to enable standard mode, slew rate control is enabled - use with Chip_IOCON_PinMux

4.16.3 Function Documentation

4.16.3.1 STATIC INLINE void Chip_IOCON_Init (LPC_IOCON_T * *pIOCON*)

Initialize the IOCON peripheral.

Parameters

pIOCON	: The base of IOCON peripheral on the chip

Returns

Nothing

4.16.3.2 STATIC INLINE void Chip_IOCON_PinMux (LPC_IOCON_T * *pIOCON,* uint8_t *port,* uint8_t *pin,* uint32_t *mode,* uint8_t *func*)

Setup pin modes and function.

Parameters

pIOCON	: The base of IOCON peripheral on the chip
port	: port number
pin	: gpio pin number
mode	: OR'ed values or type IOCON_*
func	: Pin function, value of type IOCON_FUNC0 to IOCON_FUNC7

Returns

Nothing

4.16.3.3 STATIC INLINE void Chip_IOCON_PinMuxSet (LPC_IOCON_T * *pIOCON,* uint8_t *port,* uint8_t *pin,* uint32_t *modefunc*)

Sets I/O Control pin mux.

Parameters

pIOCON	: The base of IOCON peripheral on the chip
port	: GPIO port to mux
pin	: GPIO pin to mux
modefunc	: OR'ed values or type IOCON_*

Returns

Nothing

4.16.3.4 void Chip_IOCON_SetPinMuxing (LPC_IOCON_T * *pIOCON,* const PINMUX_GRP_T * *pinArray,* uint32_t *arrayLength*)

Set all I/O Control pin muxing.

Parameters

pIOCON	: The base of IOCON peripheral on the chip
pinArray	: Pointer to array of pin mux selections
arrayLength	: Number of entries in pinArray

Returns

Nothing

4.17 : Common LCD functions

Functions

- void LCD_Init (void)

  Make initialization for LCD.
- void LCD_WriteChar (char data, int x, int y)

  Write single char on LCD.
- void LCD_WriteString (char ∗data, int x, int y)

  Write a string on LCD.
- void LCD_CleanDisplay (char color)

  Clean lcd's display.

4.17.1 Detailed Description

This file contains common LCD functions.

4.17.2 Function Documentation

4.17.2.1 void LCD_CleanDisplay (char *color*)

Clean lcd's display.

Parameters

color	to be on foreground

Returns

Nothing

4.17.2.2 void LCD_Init (void)

Make initialization for LCD.

Parameters

Nothing	

Returns

Nothing

Note

This initialization use SPI communication

4.17.2.3 void LCD_WriteChar (char *data,* int *x,* int *y*)

Write single char on LCD.

Parameters

data	to be written
x	initial coordenate
y	initial coordenate

Returns

Nothing

Note

This operation use SPI communication

4.17.2.4 void LCD_WriteString (char * *data*, int *x*, int *y*)

Write a string on LCD.

Parameters

data	to be written
x	initial coordenate
y	initial coordenate

Returns

Nothing

Note

This operation calls n times WriteChar till value == '\0'

4.18 BOARD: Board specific PHY drivers

Macros

- #define PHY_LINK_ERROR (1 << 0)
- #define PHY_LINK_BUSY (1 << 1)
- #define PHY_LINK_CHANGED (1 << 2)
- #define PHY_LINK_CONNECTED (1 << 3)
- #define PHY_LINK_SPEED100 (1 << 4)
- #define PHY_LINK_FULLDUPLX (1 << 5)

Functions

- uint32_t lpcPHYStsPoll (void)

  Phy status update state machine.
- uint32_t lpc_phy_init (bool rmii, p_msDelay_func_t pDelayMsFunc)

  Initialize the PHY.

4.18.1 Detailed Description

The simple PHY function API provides simple non-blocking PHY status monitoring and initialization support for various Ethernet PHYs. To initialize the PHY, call lpc_phy_init() once. lpc_phy_init() requires several standard functions from the MAC driver for interfacing to the PHY via a MII link (Chip_ENET_StartMIIWrite(), Chip_ENET_IsMIIBusy(), Chip_ENET_StartMIIRead(), and Chip_ENET_ReadMIIData()).

Once initialized, just preiodically call the lpcPHYStsPoll() function from the background loop or a thread and monitor the returned status to determine if the PHY state has changed and the current PHY state.

4.18.2 Macro Definition Documentation

4.18.2.1 #define PHY_LINK_BUSY (1 << 1)

PHY status bit for MII link busy

4.18.2.2 #define PHY_LINK_CHANGED (1 << 2)

PHY status bit for changed state (not persistent)

4.18.2.3 #define PHY_LINK_CONNECTED (1 << 3)

PHY status bit for connected state

4.18.2.4 #define PHY_LINK_ERROR (1 << 0)

PHY status bit for link error

4.18.2.5 #define PHY_LINK_FULLDUPLX (1 << 5)

PHY status bit for full duplex mode

4.18.2.6 #define PHY_LINK_SPEED100 (1 << 4)

PHY status bit for 100Mbps mode

4.18.3 Function Documentation

4.18.3.1 uint32_t lpc_phy_init (bool *rmii,* p_msDelay_func_t *pDelayMsFunc*)

Initialize the PHY.

Parameters

rmii	: Initializes PHY for RMII mode if true, MII if false
pDelayMsFunc	: Delay function (in mS) used for this driver

Returns

PHY_LINK_ERROR or 0 on success

Note

This function initializes the PHY. It will block until complete. It will not wait for the PHY to detect a connected cable and remain busy. Use lpcPHYStsPoll to detect cable insertion.

4.18.3.2 uint32_t lpcPHYStsPoll (void)

Phy status update state machine.

Returns

An Or'ed value of PHY_LINK_∗ statuses

Note

This function can be called at any rate and will poll the the PHY status. Multiple calls may be needed to determine PHY status.

4.19 CHIP: LPC Common Types

Modules

- LPC Public Types
- LPC Public Macros

4.19.1 Detailed Description

4.20 LPC Public Types

Macros

- #define **PARAM_SETSTATE**(State) ((State == RESET) || (State == SET))
- #define **PARAM_FUNCTIONALSTATE**(State) ((State == DISABLE) || (State == ENABLE))
- #define **INLINE** inline

Typedefs

- typedef enum FlagStatus **IntStatus**
- typedef enum FlagStatus **SetState**
- typedef void(* PFV)()
- typedef int32_t(* PFI)()
- typedef char CHAR
- typedef uint8_t UNS_8
- typedef int8_t INT_8
- typedef uint16_t UNS_16
- typedef int16_t INT_16
- typedef uint32_t UNS_32
- typedef int32_t INT_32
- typedef int64_t INT_64
- typedef uint64_t UNS_64
- typedef bool BOOL_32
- typedef bool BOOL_16
- typedef bool BOOL_8

Enumerations

- enum Bool { **FALSE** = 0, **TRUE** = !FALSE }

  Boolean Type definition.
- enum FlagStatus { **RESET** = 0, **SET** = !RESET }

  Boolean Type definition.
- enum FunctionalState { **DISABLE** = 0, **ENABLE** = !DISABLE }

  Functional State Definition.
- enum Status { **ERROR** = 0, **SUCCESS** = !ERROR }
- enum TRANSFER_BLOCK_T { NONE_BLOCKING = 0, BLOCKING }

4.20.1 Detailed Description

4.20.2 Typedef Documentation

4.20.2.1 typedef bool BOOL_16

16 bit boolean type

4.20.2.2 typedef bool BOOL_32

32 bit boolean type

4.20.2.3 typedef bool BOOL_8

8 bit boolean type

4.20.2.4 typedef char CHAR

LPC type for character type

4.20.2.5 typedef int16_t INT_16

LPC type for 16 bit signed value

4.20.2.6 typedef int32_t INT_32

LPC type for 32 bit signed value

4.20.2.7 typedef int64_t INT_64

LPC type for 64 bit signed value

4.20.2.8 typedef int8_t INT_8

LPC type for 8 bit signed value

4.20.2.9 typedef int32_t(∗ PFI)()

Pointer to Function returning int32_t (any number of parameters)

4.20.2.10 typedef void(∗ PFV)()

Pointer to Function returning Void (any number of parameters)

4.20.2.11 typedef uint16_t UNS_16

LPC type for 16 bit unsigned value

4.20.2.12 typedef uint32_t UNS_32

LPC type for 32 bit unsigned value

4.20.2.13 typedef uint64_t UNS_64

LPC type for 64 bit unsigned value

4.20.2.14 typedef uint8_t UNS_8

LPC type for 8 bit unsigned value

4.20.3 Enumeration Type Documentation

4.20.3.1 enum FlagStatus

Boolean Type definition.

Flag Status and Interrupt Flag Status type definition

4.20.3.2 enum Status

@ Status type definition

4.20.3.3 enum TRANSFER_BLOCK_T

Read/Write transfer type mode (Block or non-block)

Enumerator

NONE_BLOCKING None Blocking type

BLOCKING Blocking type

4.21 LPC Public Macros

Macros

- #define **_BIT**(n) (1 << (n))
- #define **_SBF**(f, v) ((v) << (f))
- #define **_BITMASK**(field_width) (_BIT(field_width) - 1)
- #define **NULL** ((void *) 0)
- #define **NELEMENTS**(array) (sizeof(array) / sizeof(array[0]))
- #define **STATIC** static
- #define **EXTERN** extern
- #define **MAX**(a, b) (((a) > (b)) ? (a) : (b))
- #define **MIN**(a, b) (((a) < (b)) ? (a) : (b))

4.21.1 Detailed Description

4.22 CHIP: LPC17XX/40XX ROM API declarations and functions

Classes

- struct LPC_ROM_API_T

  LPC17XX/40XX High level ROM API structure.

Macros

- #define **LPC_ROM_API_BASE_LOC** 0x1FFF1FF8
- #define **LPC_ROM_API** (∗(LPC_ROM_API_T ∗ ∗) LPC_ROM_API_BASE_LOC)
- #define **IAP_ENTRY_LOCATION** 0X1FFF1FF1

4.22.1 Detailed Description

4.23 CHIP: LPC17xx/40xx Real Time Clock driver

Classes

- struct LPC_RTC_T

  Real Time Clock register block structure.
- struct LPC_REGFILE_T

  Register File register block structure.
- struct RTC_TIME_T

Macros

- #define RTC_ILR_BITMASK ((0x00000003))
- #define RTC_IRL_RTCCIF ((1 << 0))
- #define RTC_IRL_RTCALF ((1 << 1))
- #define RTC_CCR_BITMASK ((0x00000013))
- #define RTC_CCR_CLKEN ((1 << 0))
- #define RTC_CCR_CTCRST ((1 << 1))
- #define RTC_CCR_CCALEN ((1 << 4))
- #define RTC_AMR_CIIR_IMSEC ((1 << 0))
- #define RTC_AMR_CIIR_IMMIN ((1 << 1))
- #define RTC_AMR_CIIR_IMHOUR ((1 << 2))
- #define RTC_AMR_CIIR_IMDOM ((1 << 3))
- #define RTC_AMR_CIIR_IMDOW ((1 << 4))
- #define RTC_AMR_CIIR_IMDOY ((1 << 5))
- #define RTC_AMR_CIIR_IMMON ((1 << 6))
- #define RTC_AMR_CIIR_IMYEAR ((1 << 7))
- #define RTC_AMR_CIIR_BITMASK ((0xFF))
- #define RTC_AUX_RTC_OSCF ((1 << 4))
- #define RTC_AUXEN_RTC_OSCFEN ((1 << 4))
- #define **RTC_CTIME0_SECONDS_MASK** ((0x3F))
- #define **RTC_CTIME0_MINUTES_MASK** ((0x3F00))
- #define **RTC_CTIME0_HOURS_MASK** ((0x1F0000))
- #define **RTC_CTIME0_DOW_MASK** ((0x7000000))
- #define **RTC_CTIME1_DOM_MASK** ((0x1F))
- #define **RTC_CTIME1_MONTH_MASK** ((0xF00))
- #define **RTC_CTIME1_YEAR_MASK** ((0xFFF0000))
- #define **RTC_CTIME2_DOY_MASK** ((0xFFF))
- #define RTC_SEC_MASK (0x0000003F)
- #define RTC_MIN_MASK (0x0000003F)
- #define RTC_HOUR_MASK (0x0000001F)
- #define RTC_DOM_MASK (0x0000001F)
- #define RTC_DOW_MASK (0x00000007)
- #define RTC_DOY_MASK (0x000001FF)
- #define RTC_MONTH_MASK (0x0000000F)
- #define RTC_YEAR_MASK (0x00000FFF)
- #define RTC_SECOND_MAX 59
- #define RTC_MINUTE_MAX 59
- #define RTC_HOUR_MAX 23
- #define RTC_MONTH_MIN 1
- #define RTC_MONTH_MAX 12
- #define RTC_DAYOFMONTH_MIN 1
- #define RTC_DAYOFMONTH_MAX 31
- #define RTC_DAYOFWEEK_MAX 6

- #define RTC_DAYOFYEAR_MIN 1
- #define RTC_DAYOFYEAR_MAX 366
- #define RTC_YEAR_MAX 4095
- #define RTC_CALIBRATION_CALVAL_MASK ((0x1FFFF))
- #define RTC_CALIBRATION_LIBDIR ((1 << 17))
- #define RTC_CALIBRATION_MAX ((0x20000))
- #define RTC_CALIB_DIR_FORWARD ((uint8_t) (0))
- #define **RTC_CALIB_DIR_BACKWARD** ((uint8_t) (1))

Enumerations

- enum RTC_TIMEINDEX_T {
  RTC_TIMETYPE_SECOND, RTC_TIMETYPE_MINUTE, RTC_TIMETYPE_HOUR, RTC_TIMETYPE_DA↩
  YOFMONTH,
  RTC_TIMETYPE_DAYOFWEEK, RTC_TIMETYPE_DAYOFYEAR, RTC_TIMETYPE_MONTH, RTC_TIM↩
  ETYPE_YEAR,
  RTC_TIMETYPE_LAST }

  RTC time type option.
- enum RTC_INT_OPT_T { RTC_INT_COUNTER_INCREASE = RTC_IRL_RTCCIF, RTC_INT_ALARM = R↩
  TC_IRL_RTCALF }

  RTC enumeration.

Functions

- void Chip_RTC_ResetClockTickCounter (LPC_RTC_T *pRTC)

  Reset clock tick counter in the RTC peripheral.
- void Chip_RTC_Enable (LPC_RTC_T *pRTC, FunctionalState NewState)

  Start/Stop RTC peripheral.
- void Chip_RTC_CntIncrIntConfig (LPC_RTC_T *pRTC, uint32_t cntrMask, FunctionalState NewState)

  Enable/Disable Counter increment interrupt for a time type in the RTC peripheral.
- void Chip_RTC_AlarmIntConfig (LPC_RTC_T *pRTC, uint32_t alarmMask, FunctionalState NewState)

  Enable/Disable Alarm interrupt for a time type in the RTC peripheral.
- STATIC INLINE void Chip_RTC_SetTime (LPC_RTC_T *pRTC, RTC_TIMEINDEX_T Timetype, uint32_↩
  t TimeValue)

  Set current time value for a time type in the RTC peripheral.
- STATIC INLINE uint32_t Chip_RTC_GetTime (LPC_RTC_T *pRTC, RTC_TIMEINDEX_T Timetype)

  Get current time value for a type time type.
- void Chip_RTC_SetFullTime (LPC_RTC_T *pRTC, RTC_TIME_T *pFullTime)

  Set full time in the RTC peripheral.
- void Chip_RTC_GetFullTime (LPC_RTC_T *pRTC, RTC_TIME_T *pFullTime)

  Get full time from the RTC peripheral.
- STATIC INLINE void Chip_RTC_SetAlarmTime (LPC_RTC_T *pRTC, RTC_TIMEINDEX_T Timetype, uint32_t ALValue)

  Set alarm time value for a time type.
- STATIC INLINE uint32_t Chip_RTC_GetAlarmTime (LPC_RTC_T *pRTC, RTC_TIMEINDEX_T Timetype)

  Get alarm time value for a time type.
- void Chip_RTC_SetFullAlarmTime (LPC_RTC_T *pRTC, RTC_TIME_T *pFullTime)

  Set full alarm time in the RTC peripheral.
- void Chip_RTC_GetFullAlarmTime (LPC_RTC_T *pRTC, RTC_TIME_T *pFullTime)

  Get full alarm time in the RTC peripheral.
- STATIC INLINE void Chip_REGFILE_Write (LPC_REGFILE_T *pRegFile, uint8_t index, uint32_t value)

  Write value to General purpose registers.

- STATIC INLINE uint32_t Chip_REGFILE_Read (LPC_REGFILE_T *pRegFile, uint8_t index)
  Read value from General purpose registers.
- void Chip_RTC_CalibCounterCmd (LPC_RTC_T *pRTC, FunctionalState NewState)
  Enable/Disable calibration counter in the RTC peripheral.
- STATIC INLINE void Chip_RTC_CalibConfig (LPC_RTC_T *pRTC, uint32_t CalibValue, uint8_t CalibDir)
  Configures Calibration in the RTC peripheral.
- STATIC INLINE void Chip_RTC_ClearIntPending (LPC_RTC_T *pRTC, uint32_t IntType)
  Clear specified Location interrupt pending in the RTC peripheral.
- STATIC INLINE IntStatus Chip_RTC_GetIntPending (LPC_RTC_T *pRTC, uint32_t IntType)
  Check whether if specified location interrupt in the RTC peripheral is set or not.
- void Chip_RTC_Init (LPC_RTC_T *pRTC)
  Initialize the RTC peripheral.
- void Chip_RTC_DeInit (LPC_RTC_T *pRTC)
  De-initialize the RTC peripheral.

4.23.1 Detailed Description

4.23.2 Macro Definition Documentation

4.23.2.1 #define RTC_AMR_CIIR_BITMASK ((0xFF))

CIIR bit mask

4.23.2.2 #define RTC_AMR_CIIR_IMDOM ((1 << 3))

Counter Increment Interrupt bit for day of month

4.23.2.3 #define RTC_AMR_CIIR_IMDOW ((1 << 4))

Counter Increment Interrupt bit for day of week

4.23.2.4 #define RTC_AMR_CIIR_IMDOY ((1 << 5))

Counter Increment Interrupt bit for day of year

4.23.2.5 #define RTC_AMR_CIIR_IMHOUR ((1 << 2))

Counter Increment Interrupt bit for hour

4.23.2.6 #define RTC_AMR_CIIR_IMMIN ((1 << 1))

Counter Increment Interrupt bit for minute

4.23.2.7 #define RTC_AMR_CIIR_IMMON ((1 << 6))

Counter Increment Interrupt bit for month

4.23.2.8 #define RTC_AMR_CIIR_IMSEC ((1 << 0))

Counter Increment Interrupt bit for second

4.23.2.9 #define RTC_AMR_CIIR_IMYEAR ((1 << 7))

Counter Increment Interrupt bit for year

4.23.2.10 #define RTC_AUX_RTC_OSCF ((1 << 4))

RTC Oscillator Fail detect flag

4.23.2.11 #define RTC_AUXEN_RTC_OSCFEN ((1 << 4))

Oscillator Fail Detect interrupt enable

4.23.2.12 #define RTC_CALIB_DIR_FORWARD ((uint8_t) (0))

Calibration definitions

4.23.2.13 #define RTC_CALIBRATION_CALVAL_MASK ((0x1FFFF))

Calibration value

4.23.2.14 #define RTC_CALIBRATION_LIBDIR ((1 << 17))

Calibration direction

4.23.2.15 #define RTC_CALIBRATION_MAX ((0x20000))

Calibration max value

4.23.2.16 #define RTC_CCR_BITMASK ((0x00000013))

CCR register mask

4.23.2.17 #define RTC_CCR_CCALEN ((1 << 4))

Calibration counter enable

4.23.2.18 #define RTC_CCR_CLKEN ((1 << 0))

Clock enable

4.23.2.19 #define RTC_CCR_CTCRST ((1 << 1))

Clock reset

4.23.2.20 #define RTC_DAYOFMONTH_MAX 31

Maximum value of day of month

4.23.2.21 #define RTC_DAYOFMONTH_MIN 1

Minimum value of day of month

4.23.2.22 #define RTC_DAYOFWEEK_MAX 6

Maximum value of day of week

4.23.2.23 #define RTC_DAYOFYEAR_MAX 366

Maximum value of day of year

4.23.2.24 #define RTC_DAYOFYEAR_MIN 1

Minimum value of day of year

4.23.2.25 #define RTC_DOM_MASK (0x0000001F)

DOM register mask

4.23.2.26 #define RTC_DOW_MASK (0x00000007)

DOW register mask

4.23.2.27 #define RTC_DOY_MASK (0x000001FF)

DOY register mask

4.23.2.28 #define RTC_HOUR_MASK (0x0000001F)

HOUR register mask

4.23.2.29 #define RTC_HOUR_MAX 23

Maximum value of hour

4.23.2.30 #define RTC_ILR_BITMASK ((0x00000003))

ILR register mask

4.23.2.31 #define RTC_IRL_RTCALF ((1 << 1))

Bit inform the source interrupt is alarm match

4.23.2.32 #define RTC_IRL_RTCCIF ((1 << 0))

Bit inform the source interrupt is counter increment

4.23.2.33 #define RTC_MIN_MASK (0x0000003F)

MIN register mask

4.23.2.34 #define RTC_MINUTE_MAX 59

Maximum value of minute

4.23.2.35 #define RTC_MONTH_MASK (0x0000000F)

MONTH register mask

4.23.2.36 #define RTC_MONTH_MAX 12

Maximum value of month

4.23.2.37 #define RTC_MONTH_MIN 1

Minimum value of month

4.23.2.38 #define RTC_SEC_MASK (0x0000003F)

SEC register mask

4.23.2.39 #define RTC_SECOND_MAX 59

Maximum value of second

4.23.2.40 #define RTC_YEAR_MASK (0x00000FFF)

YEAR register mask

4.23.2.41 #define RTC_YEAR_MAX 4095

Maximum value of year

4.23.3 Enumeration Type Documentation

4.23.3.1 enum RTC_INT_OPT_T

RTC enumeration.

RTC interrupt source

Enumerator

RTC_INT_COUNTER_INCREASE Counter Increment Interrupt

RTC_INT_ALARM The alarm interrupt

4.23.3.2 enum RTC_TIMEINDEX_T

RTC time type option.

Enumerator

RTC_TIMETYPE_SECOND Second

RTC_TIMETYPE_MINUTE Month

RTC_TIMETYPE_HOUR Hour

RTC_TIMETYPE_DAYOFMONTH Day of month

RTC_TIMETYPE_DAYOFWEEK Day of week

RTC_TIMETYPE_DAYOFYEAR Day of year

RTC_TIMETYPE_MONTH Month

RTC_TIMETYPE_YEAR Year

4.23.4 Function Documentation

4.23.4.1 STATIC INLINE uint32_t Chip_REGFILE_Read (LPC_REGFILE_T * *pRegFile*, uint8_t *index*)

Read value from General purpose registers.

Parameters

pRegFile	: RegFile peripheral selected
index	: General purpose register index

Returns

Read Value

Note

These General purpose registers can be used to store important information when the main power supply is off. The value in these registers is not affected by chip reset. These registers are powered in the RTC power domain.

4.23.4.2 STATIC INLINE void Chip_REGFILE_Write (LPC_REGFILE_T * *pRegFile*, uint8_t *index*, uint32_t *value*)

Write value to General purpose registers.

Parameters

pRegFile	: RegFile peripheral selected
index	: General purpose register index
value	: Value to write

Returns

None

Note

These General purpose registers can be used to store important information when the main power supply is off. The value in these registers is not affected by chip reset. These registers are powered in the RTC power domain.

4.23.4.3 void Chip_RTC_AlarmIntConfig (LPC_RTC_T * *pRTC,* uint32_t *alarmMask,* FunctionalState *NewState*)

Enable/Disable Alarm interrupt for a time type in the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected
alarmMask	: Or'ed bit values for ALARM types (RTC_AMR_CIIR_IM∗)
NewState	: ENABLE or DISABLE

Returns

None

4.23.4.4 STATIC INLINE void Chip_RTC_CalibConfig (LPC_RTC_T ∗ *pRTC,* uint32_t *CalibValue,* uint8_t *CalibDir*)

Configures Calibration in the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected
CalibValue	: Calibration value, should be in range from 0 to 131,072
CalibDir	: Calibration Direction, should be: • RTC_CALIB_DIR_FORWARD :Forward calibration • RTC_CALIB_DIR_BACKWARD :Backward calibration

Returns

None

4.23.4.5 void Chip_RTC_CalibCounterCmd (LPC_RTC_T ∗ *pRTC,* FunctionalState *NewState*)

Enable/Disable calibration counter in the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected
NewState	: New State of this function, should be: • ENABLE :The calibration counter is enabled and counting • DISABLE :The calibration counter is disabled and reset to zero

Returns

None

4.23.4.6 STATIC INLINE void Chip_RTC_ClearIntPending (LPC_RTC_T ∗ *pRTC,* uint32_t *IntType*)

Clear specified Location interrupt pending in the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected

IntType	: Interrupt location type, should be: • RTC_INT_COUNTER_INCREASE :Clear Counter Increment Interrupt pending. • RTC_INT_ALARM :Clear alarm interrupt pending

Returns

None

4.23.4.7 void Chip_RTC_CntIncrIntConfig (LPC_RTC_T * *pRTC,* uint32_t *cntrMask,* FunctionalState *NewState*)

Enable/Disable Counter increment interrupt for a time type in the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected
cntrMask	: Or'ed bit values for time types (RTC_AMR_CIIR_IM*)
NewState	: ENABLE or DISABLE

Returns

None

4.23.4.8 void Chip_RTC_DeInit (LPC_RTC_T * *pRTC*)

De-initialize the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected

Returns

None

4.23.4.9 void Chip_RTC_Enable (LPC_RTC_T * *pRTC,* FunctionalState *NewState*)

Start/Stop RTC peripheral.

Parameters

pRTC	: RTC peripheral selected
NewState	: New State of this function, should be: • ENABLE :The time counters are enabled • DISABLE :The time counters are disabled

Returns

None

4.23.4.10 STATIC INLINE uint32_t Chip_RTC_GetAlarmTime (LPC_RTC_T * *pRTC*, RTC_TIMEINDEX_T *Timetype*)

Get alarm time value for a time type.

Parameters

pRTC	: RTC peripheral selected
Timetype	: Time index field to get

Returns

Value of Alarm time according to specified time type

4.23.4.11 void Chip_RTC_GetFullAlarmTime (LPC_RTC_T * *pRTC*, RTC_TIME_T * *pFullTime*)

Get full alarm time in the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected
pFullTime	: Pointer to full time record to fill

Returns

None

4.23.4.12 void Chip_RTC_GetFullTime (LPC_RTC_T * *pRTC*, RTC_TIME_T * *pFullTime*)

Get full time from the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected
pFullTime	: Pointer to full time record to fill

Returns

None

4.23.4.13 STATIC INLINE IntStatus Chip_RTC_GetIntPending (LPC_RTC_T * *pRTC*, uint32_t *IntType*)

Check whether if specified location interrupt in the RTC peripheral is set or not.

Parameters

pRTC	: RTC peripheral selected
IntType	: Interrupt location type, should be: • RTC_INT_COUNTER_INCREASE: Counter Increment Interrupt block generated an interrupt. • RTC_INT_ALARM: Alarm generated an interrupt.

Returns

New state of specified Location interrupt in RTC peripheral, SET OR RESET

4.23.4.14 STATIC INLINE uint32_t Chip_RTC_GetTime (LPC_RTC_T * *pRTC,* RTC_TIMEINDEX_T *Timetype*)

Get current time value for a type time type.

Parameters

pRTC	: RTC peripheral selected
Timetype	: Time field index type to get

Returns

Value of time field according to specified time type

4.23.4.15 void Chip_RTC_Init (LPC_RTC_T * *pRTC*)

Initialize the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected

Returns

None

4.23.4.16 void Chip_RTC_ResetClockTickCounter (LPC_RTC_T * *pRTC*)

Reset clock tick counter in the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected

Returns

None

4.23.4.17 STATIC INLINE void Chip_RTC_SetAlarmTime (LPC_RTC_T * *pRTC,* RTC_TIMEINDEX_T *Timetype,* uint32_t *ALValue*)

Set alarm time value for a time type.

Parameters

pRTC	: RTC peripheral selected
Timetype	: Time index field to set
ALValue	: Alarm time value to set

Returns

None

4.23.4.18 void Chip_RTC_SetFullAlarmTime (LPC_RTC_T * *pRTC*, RTC_TIME_T * *pFullTime*)

Set full alarm time in the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected
pFullTime	: Pointer to full time record to set alarm

Returns

None

4.23.4.19 void Chip_RTC_SetFullTime (LPC_RTC_T * *pRTC*, RTC_TIME_T * *pFullTime*)

Set full time in the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected
pFullTime	: Pointer to full time data

Returns

None

4.23.4.20 STATIC INLINE void Chip_RTC_SetTime (LPC_RTC_T * *pRTC*, RTC_TIMEINDEX_T *Timetype*, uint32_t *TimeValue*)

Set current time value for a time type in the RTC peripheral.

Parameters

pRTC	: RTC peripheral selected
Timetype	: time field index type to set
TimeValue	: Value to palce in time field

Returns

None

4.24 : Common SPI functions

Functions

- void SPI_Init (void)

  Make initialization for SPI communication.
- void SPI_Transfer (char data, char DnC)

  Transfer data via SPI communication.

4.24.1 Detailed Description

This file contains common SPI functions.

4.24.2 Function Documentation

4.24.2.1 void SPI_Init (void)

Make initialization for SPI communication.

Parameters

Nothing	

Returns

Nothing

4.24.2.2 void SPI_Transfer (char *data*, char *DnC*)

Transfer data via SPI communication.

Parameters

data	to be written
data	type, data=1 or command=0

Returns

Nothing

4.25 CHIP: LPC17xx/40xx System Control block driver

Classes

- struct SYSCTL_PLL_REGS_T

  LPC17XX/40XX Clock and Power PLL register block structure.
- struct LPC_SYSCTL_T

  LPC17XX/40XX Clock and Power register block structure.

Macros

- #define SYSCTL_RST_POR (1 << 0)
- #define SYSCTL_RST_EXTRST (1 << 1)
- #define SYSCTL_RST_WDT (1 << 2)
- #define SYSCTL_RST_BOD (1 << 3)
- #define SYSCTL_PD_SMFLAG (1 << 8)
- #define SYSCTL_PD_DSFLAG (1 << 9)
- #define SYSCTL_PD_PDFLAG (1 << 10)
- #define SYSCTL_PD_DPDFLAG (1 << 11)

Typedefs

- typedef enum CHIP_SYSCTL_BOOT_MODE_REMAP CHIP_SYSCTL_BOOT_MODE_REMAP_T

Enumerations

- enum CHIP_SYSCTL_PLL_T { SYSCTL_MAIN_PLL, SYSCTL_USB_PLL }
- enum FMC_FLASHTIM_T {
  FLASHTIM_20MHZ_CPU = 0, FLASHTIM_40MHZ_CPU = 1, FLASHTIM_60MHZ_CPU = 2, FLASHTIM_80MHZ_CPU = 3,
  FLASHTIM_100MHZ_CPU = 4, FLASHTIM_120MHZ_CPU = 4, FLASHTIM_SAFE_SETTING = 5 }

  FLASH Access time definitions.
- enum CHIP_SYSCTL_BOOT_MODE_REMAP { REMAP_BOOT_LOADER_MODE, REMAP_USER_FLASH_MODE }
- enum CHIP_SYSCTL_RESET_T {
  SYSCTL_RESET_LCD, SYSCTL_RESET_TIMER0, SYSCTL_RESET_TIMER1, SYSCTL_RESET_UART0,
  SYSCTL_RESET_UART1, SYSCTL_RESET_PWM0, SYSCTL_RESET_PWM1, SYSCTL_RESET_I2C0,
  SYSCTL_RESET_UART4, SYSCTL_RESET_RTC, SYSCTL_RESET_SSP1, SYSCTL_RESET_EMC,
  SYSCTL_RESET_ADC, SYSCTL_RESET_CAN1, SYSCTL_RESET_CAN2, SYSCTL_RESET_GPIO,
  SYSCTL_RESET_SPIFI, SYSCTL_RESET_MCPWM, SYSCTL_RESET_QEI, SYSCTL_RESET_I2C1,
  SYSCTL_RESET_SSP2, SYSCTL_RESET_SSP0, SYSCTL_RESET_TIMER2, SYSCTL_RESET_TIMER3,
  SYSCTL_RESET_UART2, SYSCTL_RESET_UART3, SYSCTL_RESET_I2C2, SYSCTL_RESET_I2S,
  SYSCTL_RESET_PCSDC, SYSCTL_RESET_GPDMA, SYSCTL_RESET_ENET, SYSCTL_RESET_USB,
  SYSCTL_RESET_IOCON, SYSCTL_RESET_DAC, SYSCTL_RESET_CANACC }

Functions

- STATIC INLINE void Chip_SYSCTL_SetFLASHAccess (FMC_FLASHTIM_T clks)

  Set FLASH memory access time in clocks.
- STATIC INLINE void Chip_SYSCTL_Map (CHIP_SYSCTL_BOOT_MODE_REMAP_T remap)

  Re-map interrupt vectors.

- STATIC INLINE uint32_t Chip_SYSCTL_GetSystemRSTStatus (void)

  Get system reset status.
- STATIC INLINE void Chip_SYSCTL_ClearSystemRSTStatus (uint32_t reset)

  Clear system reset status.
- STATIC INLINE void Chip_SYSCTL_EnableBOD (void)

  Enable brown-out detection.
- STATIC INLINE void Chip_SYSCTL_DisableBOD (void)

  Disable brown-out detection.
- STATIC INLINE void Chip_SYSCTL_EnableBODReset (void)

  Enable brown-out detection reset.
- STATIC INLINE void Chip_SYSCTL_DisableBODReset (void)

  Disable brown-out detection reset.
- STATIC INLINE void Chip_SYSCTL_EnableBODRPM (void)

  Enable brown-out detection reduced power mode.
- STATIC INLINE void Chip_SYSCTL_DisableBODRPM (void)

  Disable brown-out detection reduced power mode.
- uint32_t Chip_SYSCTL_GetClrSleepFlags (uint32_t flags)

  Returns and clears the current sleep mode entry flags.
- STATIC INLINE void Chip_SYSCTL_EnableBoost (void)

  Enable power boost for clock operation over 100MHz.
- STATIC INLINE void Chip_SYSCTL_DisableBoost (void)

  Disable power boost for clock operation under 100MHz.
- void Chip_SYSCTL_PeriphReset (CHIP_SYSCTL_RESET_T periph)

  Resets a peripheral.

4.25.1 Detailed Description

4.25.2 Macro Definition Documentation

4.25.2.1 #define SYSCTL_PD_DPDFLAG (1 << 11)

Deep Power-down entry flag

4.25.2.2 #define SYSCTL_PD_DSFLAG (1 << 9)

Deep Sleep entry flag

4.25.2.3 #define SYSCTL_PD_PDFLAG (1 << 10)

Power-down entry flag

4.25.2.4 #define SYSCTL_PD_SMFLAG (1 << 8)

Sleep Mode entry flag

4.25.2.5 #define SYSCTL_RST_BOD (1 << 3)

Brown-out detect reset status

4.25.2.6 #define SYSCTL_RST_EXTRST (1 << 1)

External reset status

4.25.2.7 #define SYSCTL_RST_POR (1 << 0)

System reset statusPOR reset status

4.25.2.8 #define SYSCTL_RST_WDT (1 << 2)

Watchdog reset status

4.25.3 Typedef Documentation

4.25.3.1 typedef enum CHIP_SYSCTL_BOOT_MODE_REMAP CHIP_SYSCTL_BOOT_MODE_REMAP_T

System memory remap modes used to remap interrupt vectors

4.25.4 Enumeration Type Documentation

4.25.4.1 enum CHIP_SYSCTL_BOOT_MODE_REMAP

System memory remap modes used to remap interrupt vectors

Enumerator

REMAP_BOOT_LOADER_MODE Interrupt vectors are re-mapped to Boot ROM

REMAP_USER_FLASH_MODE Interrupt vectors are not re-mapped and reside in Flash

4.25.4.2 enum CHIP_SYSCTL_PLL_T

Selectable PLLs

Enumerator

SYSCTL_MAIN_PLL Main PLL (PLL0)

SYSCTL_USB_PLL USB PLL (PLL1)

4.25.4.3 enum CHIP_SYSCTL_RESET_T

Peripheral reset numbers This is a list of peripherals that can be reset

Enumerator

SYSCTL_RESET_LCD LCD reset

SYSCTL_RESET_TIMER0 Timer 0 reset

SYSCTL_RESET_TIMER1 Timer 1 reset

SYSCTL_RESET_UART0 UART 0 reset

SYSCTL_RESET_UART1 UART 1 reset

SYSCTL_RESET_PWM0 PWM0 reset

SYSCTL_RESET_PWM1 PWM1 reset

SYSCTL_RESET_I2C0 I2C0 reset
SYSCTL_RESET_UART4 UART 4 reset
SYSCTL_RESET_RTC RTC reset
SYSCTL_RESET_SSP1 SSP1 reset
SYSCTL_RESET_EMC EMC reset
SYSCTL_RESET_ADC ADC reset
SYSCTL_RESET_CAN1 CAN1 reset
SYSCTL_RESET_CAN2 CAN2 reset
SYSCTL_RESET_GPIO GPIO reset
SYSCTL_RESET_SPIFI SPIFI reset
SYSCTL_RESET_MCPWM MCPWM reset
SYSCTL_RESET_QEI QEI reset
SYSCTL_RESET_I2C1 I2C1 reset
SYSCTL_RESET_SSP2 SSP2 reset
SYSCTL_RESET_SSP0 SSP0 reset
SYSCTL_RESET_TIMER2 Timer 2 reset
SYSCTL_RESET_TIMER3 Timer 3 reset
SYSCTL_RESET_UART2 UART 2 reset
SYSCTL_RESET_UART3 UART 3 reset
SYSCTL_RESET_I2C2 I2C2 reset
SYSCTL_RESET_I2S I2S reset
SYSCTL_RESET_PCSDC SD Card interface reset
SYSCTL_RESET_GPDMA GP DMA reset
SYSCTL_RESET_ENET EMAC/Ethernet reset
SYSCTL_RESET_USB USB reset
SYSCTL_RESET_IOCON IOCON reset
SYSCTL_RESET_DAC DAC reset
SYSCTL_RESET_CANACC CAN acceptance filter reset

4.25.4.4 enum FMC_FLASHTIM_T

FLASH Access time definitions.

Enumerator

FLASHTIM_20MHZ_CPU Flash accesses use 1 CPU clocks. Use for up to 20 MHz CPU clock
FLASHTIM_40MHZ_CPU Flash accesses use 2 CPU clocks. Use for up to 40 MHz CPU clock
FLASHTIM_60MHZ_CPU Flash accesses use 3 CPU clocks. Use for up to 60 MHz CPU clock
FLASHTIM_80MHZ_CPU Flash accesses use 4 CPU clocks. Use for up to 80 MHz CPU clock
FLASHTIM_100MHZ_CPU Flash accesses use 5 CPU clocks. Use for up to 100 MHz CPU clock
FLASHTIM_120MHZ_CPU Flash accesses use 5 CPU clocks. Use for up to 120 Mhz for LPC1759 and LPC1769 only.
FLASHTIM_SAFE_SETTING Flash accesses use 6 CPU clocks. Safe setting for any allowed conditions

4.25.5 Function Documentation

4.25.5.1 STATIC INLINE void Chip_SYSCTL_ClearSystemRSTStatus (uint32_t *reset*)

Clear system reset status.

Parameters

reset	: An Or'ed value of SYSCTL_RST_∗ status to clear

Returns

Nothing

4.25.5.2 STATIC INLINE void Chip_SYSCTL_DisableBOD (void)

Disable brown-out detection.

Returns

Nothing

4.25.5.3 STATIC INLINE void Chip_SYSCTL_DisableBODReset (void)

Disable brown-out detection reset.

Returns

Nothing

4.25.5.4 STATIC INLINE void Chip_SYSCTL_DisableBODRPM (void)

Disable brown-out detection reduced power mode.

Returns

Nothing

4.25.5.5 STATIC INLINE void Chip_SYSCTL_DisableBoost (void)

Disable power boost for clock operation under 100MHz.

Returns

Nothing

4.25.5.6 STATIC INLINE void Chip_SYSCTL_EnableBOD (void)

Enable brown-out detection.

Returns

Nothing

4.25.5.7 STATIC INLINE void Chip_SYSCTL_EnableBODReset (void)

Enable brown-out detection reset.

Returns

Nothing

4.25.5.8 STATIC INLINE void Chip_SYSCTL_EnableBODRPM (void)

Enable brown-out detection reduced power mode.

Returns

Nothing

4.25.5.9 STATIC INLINE void Chip_SYSCTL_EnableBoost (void)

Enable power boost for clock operation over 100MHz.

Returns

Nothing

4.25.5.10 uint32_t Chip_SYSCTL_GetClrSleepFlags (uint32_t *flags*)

Returns and clears the current sleep mode entry flags.

Parameters

flags	One or more flags to clear, SYSCTL_PD_*

Returns

An Or'ed value of the sleep flags, SYSCTL_PD_*

Note

These flags indicate the successful entry of one or more sleep modes.

4.25.5.11 STATIC INLINE uint32_t Chip_SYSCTL_GetSystemRSTStatus (void)

Get system reset status.

Returns

An Or'ed value of SYSCTL_RST_*

Note

This function returns the detected reset source(s).

4.25.5.12 STATIC INLINE void Chip_SYSCTL_Map (CHIP_SYSCTL_BOOT_MODE_REMAP_T *remap*)

Re-map interrupt vectors.

Parameters

remap	: system memory map value

Returns

Nothing

4.25.5.13 void Chip_SYSCTL_PeriphReset (CHIP_SYSCTL_RESET_T *periph*)

Resets a peripheral.

Parameters

periph	Peripheral to reset

Returns

Nothing

4.25.5.14 STATIC INLINE void Chip_SYSCTL_SetFLASHAccess (FMC_FLASHTIM_T *clks*)

Set FLASH memory access time in clocks.

Parameters

clks	: Clock cycles for FLASH access (minus 1)

Returns

Nothing

Note

See the user manual for valid settings for this register for when power boot is enabled or off.

4.26 : Common Radio functions

Functions

- void TEA5767_Init (unsigned int freq)
  Makes radio initialization.
- unsigned TEA5767_FMToPLL (unsigned int freq)
  convert fm frequency to PLL value
- unsigned TEA5767_PLLToFM (unsigned int pll)
  Convert pll value to frequency value.
- void TEA5767_SearchUp (void)
  Search up.
- void TEA5767_SearchDown (void)
  Search down.
- unsigned int TEA5767_GetFrequency (void)
  Get current frequency.
- unsigned int TEA5767_GetFrequencyInPLL (void)
  Get current frequency in pll value.
- void TEA5767_SetFrequency (unsigned int pllVal)
  set radio current frequency

4.26.1 Detailed Description

This file contains common Radio functions.

4.26.2 Function Documentation

4.26.2.1 unsigned TEA5767_FMToPLL (unsigned int *freq*)

convert fm frequency to PLL value

Parameters

frequency	

Returns

value in fm frequency

4.26.2.2 unsigned int TEA5767_GetFrequency (void)

Get current frequency.

Parameters

Nothing	

Returns

Nothing

4.26.2.3 unsigned int TEA5767_GetFrequencyInPLL (void)

Get current frequency in pll value.

Parameters

Nothing	

Returns

Nothing

4.26.2.4 void TEA5767_Init (unsigned int *freq*)

Makes radio initialization.

Parameters

search	frequency

Returns

Nothing

4.26.2.5 unsigned TEA5767_PLLToFM (unsigned int *pll*)

Convert pll value to frequency value.

Parameters

pll	value

Returns

value in pll

4.26.2.6 void TEA5767_SearchDown (void)

Search down.

Parameters

Nothing	

Returns

Nothing

4.26.2.7 void TEA5767_SearchUp (void)

Search up.

Parameters

Nothing	

Returns

Nothing

4.26.2.8 void TEA5767_SetFrequency (unsigned int *pllVal*)

set radio current frequency

Parameters

frequency	in pll

Returns

Nothing

4.27 Uipopt

Files

- file uip-conf.h

Project-specific configuration options

uIP has a number of configuration options that can be overridden for each project. These are kept in a project-specific uip-conf.h file and all configuration names have the prefix UIP_CONF.

- typedef uint8_t u8_t
- typedef uint16_t u16_t
- typedef unsigned short uip_stats_t
- #define UIP_CONF_MAX_CONNECTIONS
- #define UIP_CONF_MAX_LISTENPORTS
- #define UIP_CONF_BUFFER_SIZE
- #define UIP_CONF_BYTE_ORDER
- #define UIP_CONF_LOGGING
- #define UIP_CONF_UDP
- #define UIP_CONF_UDP_CHECKSUMS
- #define UIP_CONF_STATISTICS

4.27.1 Detailed Description

4.27.2 Macro Definition Documentation

4.27.2.1 #define UIP_CONF_BUFFER_SIZE

uIP buffer size.

4.27.2.2 #define UIP_CONF_BYTE_ORDER

CPU byte order.

4.27.2.3 #define UIP_CONF_LOGGING

Logging on or off

4.27.2.4 #define UIP_CONF_MAX_CONNECTIONS

Maximum number of TCP connections.

4.27.2.5 #define UIP_CONF_MAX_LISTENPORTS

Maximum number of listening TCP ports.

4.27.2.6 #define UIP_CONF_STATISTICS

uIP statistics on or off

4.27.2.7 #define UIP_CONF_UDP

UDP support on or off

4.27.2.8 #define UIP_CONF_UDP_CHECKSUMS

UDP checksums on or off

4.27.3 Typedef Documentation

4.27.3.1 typedef uint16_t u16_t

16 bit datatype

This typedef defines the 16-bit type used throughout uIP.

4.27.3.2 typedef uint8_t u8_t

8 bit datatype

This typedef defines the 8-bit type used throughout uIP.

4.27.3.3 typedef unsigned short uip_stats_t

Statistics datatype

This typedef defines the dataype used for keeping statistics in uIP.

Chapter 5

Class Documentation

5.1 _sations Struct Reference

Stations struct to be used.

`#include <SE2_specific.h>`

Public Attributes

- unsigned int **frequency** [STATIONS_MAX_MEM]
- unsigned int **valid_information**

5.1.1 Detailed Description

Stations struct to be used.

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

- SE2_specific.h

5.2 ENET_CONTROL_T Struct Reference

Ethernet Control register block structure.

`#include <enet_17xx_40xx.h>`

Public Attributes

- __IO uint32_t COMMAND
- __I uint32_t STATUS
- ENET_TRANSFER_INFO_T RX
- ENET_TRANSFER_INFO_T TX
- uint32_t **RESERVED0** [10]
- __I uint32_t TSV0
- __I uint32_t TSV1
- __I uint32_t RSV
- uint32_t **RESERVED1** [3]
- __IO uint32_t FLOWCONTROLCOUNTER
- __I uint32_t FLOWCONTROLSTATUS

5.2.1 Detailed Description

Ethernet Control register block structure.

5.2.2 Member Data Documentation

5.2.2.1 __IO uint32_t ENET_CONTROL_T::COMMAND

Command register

5.2.2.2 __IO uint32_t ENET_CONTROL_T::FLOWCONTROLCOUNTER

Flow control counter register

5.2.2.3 __I uint32_t ENET_CONTROL_T::FLOWCONTROLSTATUS

Flow control status register

5.2.2.4 __I uint32_t ENET_CONTROL_T::RSV

Receive status vector register

5.2.2.5 ENET_TRANSFER_INFO_T ENET_CONTROL_T::RX

Receive block registers

5.2.2.6 __I uint32_t ENET_CONTROL_T::STATUS

Status register

5.2.2.7 __I uint32_t ENET_CONTROL_T::TSV0

Transmit status vector 0 register

5.2.2.8 __I uint32_t ENET_CONTROL_T::TSV1

Transmit status vector 1 register

5.2.2.9 ENET_TRANSFER_INFO_T ENET_CONTROL_T::TX

Transmit block registers

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

- enet_17xx_40xx.h

5.3 ENET_MAC_T Struct Reference

Ethernet MAC register block structure.

`#include <enet_17xx_40xx.h>`

Public Attributes

- __IO uint32_t MAC1
- __IO uint32_t MAC2
- __IO uint32_t IPGT
- __IO uint32_t IPGR
- __IO uint32_t CLRT
- __IO uint32_t MAXF
- __IO uint32_t SUPP
- __IO uint32_t TEST
- __IO uint32_t MCFG
- __IO uint32_t MCMD
- __IO uint32_t MADR
- __O uint32_t MWTD
- __I uint32_t MRDD
- __I uint32_t MIND
- uint32_t **RESERVED0** [2]
- __IO uint32_t SA [3]

5.3.1 Detailed Description

Ethernet MAC register block structure.

5.3.2 Member Data Documentation

5.3.2.1 __IO uint32_t ENET_MAC_T::CLRT

Collision window / Retry register

5.3.2.2 __IO uint32_t ENET_MAC_T::IPGR

Non Back-to-Back Inter-Packet-Gap register

5.3.2.3 __IO uint32_t ENET_MAC_T::IPGT

Back-to-Back Inter-Packet-Gap register

5.3.2.4 __IO uint32_t ENET_MAC_T::MAC1

MAC Configuration register 1

5.3.2.5 __IO uint32_t ENET_MAC_T::MAC2

MAC Configuration register 2

5.3.2.6 __IO uint32_t ENET_MAC_T::MADR

MII Mgmt Address register

5.3.2.7 __IO uint32_t ENET_MAC_T::MAXF

Maximum Frame register

5.3.2.8 __IO uint32_t ENET_MAC_T::MCFG

MII Mgmt Configuration register

5.3.2.9 __IO uint32_t ENET_MAC_T::MCMD

MII Mgmt Command register

5.3.2.10 __I uint32_t ENET_MAC_T::MIND

MII Mgmt Indicators register

5.3.2.11 __I uint32_t ENET_MAC_T::MRDD

MII Mgmt Read Data register

5.3.2.12 __O uint32_t ENET_MAC_T::MWTD

MII Mgmt Write Data register

5.3.2.13 __IO uint32_t ENET_MAC_T::SA[3]

Station Address registers

5.3.2.14 __IO uint32_t ENET_MAC_T::SUPP

PHY Support register

5.3.2.15 __IO uint32_t ENET_MAC_T::TEST

Test register

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

- enet_17xx_40xx.h

5.4 ENET_MODULE_CTRL_T Struct Reference

Ethernet Module Control register block structure.

`#include <enet_17xx_40xx.h>`

Public Attributes

- __I uint32_t INTSTATUS
- __IO uint32_t INTENABLE

- __O uint32_t INTCLEAR
- __O uint32_t INTSET
- uint32_t **RESERVED**
- __IO uint32_t POWERDOWN

5.4.1 Detailed Description

Ethernet Module Control register block structure.

5.4.2 Member Data Documentation

5.4.2.1 __O uint32_t ENET_MODULE_CTRL_T::INTCLEAR

Interrupt clear register

5.4.2.2 __IO uint32_t ENET_MODULE_CTRL_T::INTENABLE

Interrupt enable register

5.4.2.3 __O uint32_t ENET_MODULE_CTRL_T::INTSET

Interrupt set register

5.4.2.4 __I uint32_t ENET_MODULE_CTRL_T::INTSTATUS

Interrupt status register

5.4.2.5 __IO uint32_t ENET_MODULE_CTRL_T::POWERDOWN

Power-down register

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

- enet_17xx_40xx.h

5.5 ENET_RXDESC_T Struct Reference

RX Descriptor structure.

`#include <enet_17xx_40xx.h>`

Public Attributes

- uint32_t Packet
- uint32_t Control

5.5.1 Detailed Description

RX Descriptor structure.

5.5.2 Member Data Documentation

5.5.2.1 uint32_t ENET_RXDESC_T::Control

Control information

5.5.2.2 uint32_t ENET_RXDESC_T::Packet

Base address of the data buffer for storing receive data

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

- enet_17xx_40xx.h

5.6 ENET_RXFILTER_T Struct Reference

Ethernet Receive Filter register block structure.

`#include <enet_17xx_40xx.h>`

Public Attributes

- __IO uint32_t CONTROL
- __I uint32_t WOLSTATUS
- __O uint32_t WOLCLEAR
- uint32_t **RESERVED**
- __IO uint32_t HashFilterL
- __IO uint32_t HashFilterH

5.6.1 Detailed Description

Ethernet Receive Filter register block structure.

5.6.2 Member Data Documentation

5.6.2.1 __IO uint32_t ENET_RXFILTER_T::CONTROL

Receive filter control register

5.6.2.2 __IO uint32_t ENET_RXFILTER_T::HashFilterH

Hash filter table MSBs register

5.6.2.3 __IO uint32_t ENET_RXFILTER_T::HashFilterL

Hash filter table LSBs register

5.6.2.4 __O uint32_t ENET_RXFILTER_T::WOLCLEAR

Receive filter WoL clear register

5.6.2.5 __I uint32_t ENET_RXFILTER_T::WOLSTATUS

Receive filter WoL status register

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

- enet_17xx_40xx.h

5.7 ENET_RXSTAT_T Struct Reference

RX Status structure.

`#include <enet_17xx_40xx.h>`

Public Attributes

- uint32_t StatusInfo
- uint32_t StatusHashCRC

5.7.1 Detailed Description

RX Status structure.

5.7.2 Member Data Documentation

5.7.2.1 uint32_t ENET_RXSTAT_T::StatusHashCRC

The concatenation of the destination address hash CRC and the source address hash CRC

5.7.2.2 uint32_t ENET_RXSTAT_T::StatusInfo

Receive status return flags.

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

- enet_17xx_40xx.h

5.8 ENET_TRANSFER_INFO_T Struct Reference

Ethernet Transfer register Block Structure.

`#include <enet_17xx_40xx.h>`

Public Attributes

- __IO uint32_t DESCRIPTOR
- __IO uint32_t STATUS
- __IO uint32_t DESCRIPTORNUMBER
- __IO uint32_t PRODUCEINDEX
- __IO uint32_t CONSUMEINDEX

5.8.1 Detailed Description

Ethernet Transfer register Block Structure.

5.8.2 Member Data Documentation

5.8.2.1 __IO uint32_t ENET_TRANSFER_INFO_T::CONSUMEINDEX

Consume index register

5.8.2.2 __IO uint32_t ENET_TRANSFER_INFO_T::DESCRIPTOR

Descriptor base address register

5.8.2.3 __IO uint32_t ENET_TRANSFER_INFO_T::DESCRIPTORNUMBER

Number of descriptors register

5.8.2.4 __IO uint32_t ENET_TRANSFER_INFO_T::PRODUCEINDEX

Produce index register

5.8.2.5 __IO uint32_t ENET_TRANSFER_INFO_T::STATUS

Status base address register

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

- enet_17xx_40xx.h

5.9 ENET_TXDESC_T Struct Reference

TX Descriptor structure.

`#include <enet_17xx_40xx.h>`

Public Attributes

- uint32_t Packet
- uint32_t Control

5.9.1 Detailed Description

TX Descriptor structure.

5.9.2 Member Data Documentation

5.9.2.1 uint32_t ENET_TXDESC_T::Control

Control information

5.9.2.2 uint32_t ENET_TXDESC_T::Packet

Base address of the data buffer containing transmit data

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

- enet_17xx_40xx.h

5.10 ENET_TXSTAT_T Struct Reference

TX Status structure.

`#include <enet_17xx_40xx.h>`

Public Attributes

- uint32_t StatusInfo

5.10.1 Detailed Description

TX Status structure.

5.10.2 Member Data Documentation

5.10.2.1 uint32_t ENET_TXSTAT_T::StatusInfo

Receive status return flags.

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

- enet_17xx_40xx.h

5.11 LPC1769_GPIO Struct Reference

GPIO registers struct.

`#include <gpio.h>`

Public Attributes

- LPC1769_Reg **FIODIR**
- LPC1769_Reg **dummy** [3]
- LPC1769_Reg **FIOMASK**
- LPC1769_Reg **FIOPIN**
- LPC1769_Reg **FIOSET**
- LPC1769_Reg **FIOCLR**

5.11.1 Detailed Description

GPIO registers struct.

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

- gpio.h

5.12 LPC1769_I2C Struct Reference

I2C registers struct.

`#include <i2c.h>`

Public Attributes

- LPC1769_Reg **I2CONSET**
- LPC1769_Reg **I2STAT**
- LPC1769_Reg **I2DAT**
- LPC1769_Reg **I2ADR0**
- LPC1769_Reg **I2SCLH**
- LPC1769_Reg **I2SCLL**
- LPC1769_Reg **I2CONCLR**
- LPC1769_Reg **MMCTRL**
- LPC1769_Reg **I2ADR1**
- LPC1769_Reg **I2ADR2**
- LPC1769_Reg **I2ADR3**
- LPC1769_Reg **I2DATA_BUFFER**
- LPC1769_Reg **I2MAKS0**
- LPC1769_Reg **I2MAKS1**
- LPC1769_Reg **I2MAKS2**
- LPC1769_Reg **I2MAKS3**

5.12.1 Detailed Description

I2C registers struct.

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

- i2c.h

5.13 LPC1769_PCB Struct Reference

PCB registers struct.

`#include <pcb.h>`

Public Attributes

- LPC1769_Reg **PINSEL0**
- LPC1769_Reg **PINSEL1**
- LPC1769_Reg **PINSEL2**
- LPC1769_Reg **PINSEL3**
- LPC1769_Reg **PINSEL4**
- LPC1769_Reg **PINSEL5**
- LPC1769_Reg **PINSEL6**
- LPC1769_Reg **PINSEL7**
- LPC1769_Reg **PINSEL8**
- LPC1769_Reg **PINSEL9**
- LPC1769_Reg **PINSEL10**
- LPC1769_Reg **PINMODE0**

- LPC1769_Reg **PINMODE1**
- LPC1769_Reg **PINMODE2**
- LPC1769_Reg **PINMODE3**
- LPC1769_Reg **PINMODE4**
- LPC1769_Reg **PINMODE5**
- LPC1769_Reg **PINMODE6**
- LPC1769_Reg **PINMODE7**
- LPC1769_Reg **PINMODE8**
- LPC1769_Reg **PINMODE9**
- LPC1769_Reg **dummy** [5]
- LPC1769_Reg **I2CPADCFG**

5.13.1 Detailed Description

PCB registers struct.

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

- pcb.h

5.14 LPC1769_PINMODE Struct Reference

PINMODE registers struct.

`#include <pcb.h>`

Public Attributes

- LPC1769_Reg **PINMODE0**
- LPC1769_Reg **PINMODE1**
- LPC1769_Reg **PINMODE2**
- LPC1769_Reg **PINMODE3**
- LPC1769_Reg **PINMODE4**
- LPC1769_Reg **PINMODE5**
- LPC1769_Reg **PINMODE6**
- LPC1769_Reg **PINMODE7**
- LPC1769_Reg **PINMODE8**
- LPC1769_Reg **PINMODE9**

5.14.1 Detailed Description

PINMODE registers struct.

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

- pcb.h

5.15 LPC1769_SPI Struct Reference

SPI registers struct.

`#include <spi.h>`

Public Attributes

- LPC1769_Reg **SOSPCR**
- LPC1769_Reg **SOSPSR**
- LPC1769_Reg **SOSPDR**
- LPC1769_Reg **SOSPCCR**
- LPC1769_Reg **RESERVED** [3]
- LPC1769_Reg **SOSPINT**

5.15.1 Detailed Description

SPI registers struct.

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

- spi.h

5.16 LPC1769_SYSTICK Struct Reference

SYSTICK registers struct.

`#include <systick.h>`

Public Attributes

- LPC1769_Reg **STCTRL**
- LPC1769_Reg **STRELOAD**
- LPC1769_Reg **STCURR**
- LPC1769_Reg **STCALIB**

5.16.1 Detailed Description

SYSTICK registers struct.

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

- systick.h

5.17 LPC_ENET_T Struct Reference

Ethernet register block structure.

`#include <enet_17xx_40xx.h>`

Public Attributes

- ENET_MAC_T MAC
- uint32_t **RESERVED1** [45]
- ENET_CONTROL_T CONTROL
- uint32_t **RESERVED4** [34]
- ENET_RXFILTER_T RXFILTER
- uint32_t **RESERVED6** [882]
- ENET_MODULE_CTRL_T MODULE_CONTROL

5.17.1 Detailed Description

Ethernet register block structure.

5.17.2 Member Data Documentation

5.17.2.1 ENET_CONTROL_T LPC_ENET_T::CONTROL

Control registers

5.17.2.2 ENET_MAC_T LPC_ENET_T::MAC

MAC registers

5.17.2.3 ENET_MODULE_CTRL_T LPC_ENET_T::MODULE_CONTROL

Module Control registers

5.17.2.4 ENET_RXFILTER_T LPC_ENET_T::RXFILTER

RxFilter registers

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

- enet_17xx_40xx.h

5.18 LPC_IOCON_T Struct Reference

IOCON register block.

`#include <iocon_17xx_40xx.h>`

Public Attributes

- __IO uint32_t **p** [5][32]

5.18.1 Detailed Description

IOCON register block.

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

- iocon_17xx_40xx.h

5.19 LPC_REGFILE_T Struct Reference

Register File register block structure.

`#include <rtc_17xx_40xx.h>`

Public Attributes

- __IO uint32_t REGFILE [5]

5.19.1 Detailed Description

Register File register block structure.

5.19.2 Member Data Documentation

5.19.2.1 __IO uint32_t LPC_REGFILE_T::REGFILE[5]

General purpose storage register

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

- rtc_17xx_40xx.h

5.20 LPC_ROM_API_T Struct Reference

LPC17XX/40XX High level ROM API structure.

`#include <romapi_17xx_40xx.h>`

Public Attributes

- const uint32_t usbdApiBase
- const uint32_t reserved0
- const uint32_t reserved1
- const uint32_t reserved2
- const uint32_t reserved3
- const uint32_t reserved4
- const uint32_t reserved5
- const uint32_t reserved6
- const uint32_t reserved7
- const uint32_t reserved8
- const uint32_t reserved9
- const uint32_t reserved10

5.20.1 Detailed Description

LPC17XX/40XX High level ROM API structure.

5.20.2 Member Data Documentation

5.20.2.1 const uint32_t LPC_ROM_API_T::reserved0

Reserved

5.20.2.2 const uint32_t LPC_ROM_API_T::reserved1

Reserved

5.20.2.3 const uint32_t LPC_ROM_API_T::reserved10

Reserved

5.20.2.4 const uint32_t LPC_ROM_API_T::reserved2

Reserved

5.20.2.5 const uint32_t LPC_ROM_API_T::reserved3

Reserved

5.20.2.6 const uint32_t LPC_ROM_API_T::reserved4

Reserved

5.20.2.7 const uint32_t LPC_ROM_API_T::reserved5

Reserved

5.20.2.8 const uint32_t LPC_ROM_API_T::reserved6

Reserved

5.20.2.9 const uint32_t LPC_ROM_API_T::reserved7

Reserved

5.20.2.10 const uint32_t LPC_ROM_API_T::reserved8

Reserved

5.20.2.11 const uint32_t LPC_ROM_API_T::reserved9

Reserved

5.20.2.12 const uint32_t LPC_ROM_API_T::usbdApiBase

USBD API function table base address

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

- romapi_17xx_40xx.h

5.21 LPC_RTC_T Struct Reference

Real Time Clock register block structure.

`#include <rtc_17xx_40xx.h>`

Public Attributes

- __IO uint32_t ILR
- __I uint32_t **RESERVED0**
- __IO uint32_t CCR
- __IO uint32_t CIIR
- __IO uint32_t AMR
- __I uint32_t CTIME [3]
- __IO uint32_t TIME [RTC_TIMETYPE_LAST]
- __IO uint32_t CALIBRATION
- __IO uint32_t GPREG [5]
- __IO uint32_t RTC_AUXEN
- __IO uint32_t RTC_AUX
- __IO uint32_t ALRM [RTC_TIMETYPE_LAST]

5.21.1 Detailed Description

Real Time Clock register block structure.

5.21.2 Member Data Documentation

5.21.2.1 __IO uint32_t LPC_RTC_T::ALRM[RTC_TIMETYPE_LAST]

Alarm field registers

5.21.2.2 __IO uint32_t LPC_RTC_T::AMR

Alarm Mask Register

5.21.2.3 __IO uint32_t LPC_RTC_T::CALIBRATION

Calibration Value Register

5.21.2.4 __IO uint32_t LPC_RTC_T::CCR

Clock Control Register

5.21.2.5 __IO uint32_t LPC_RTC_T::CIIR

Counter Increment Interrupt Register

5.21.2.6 __I uint32_t LPC_RTC_T::CTIME[3]

Consolidated Time Register 0,1,2

5.21.2.7 __IO uint32_t LPC_RTC_T::GPREG[5]

General Purpose Storage Registers

5.21.2.8 __IO uint32_t LPC_RTC_T::ILR

< RTC Structure Interrupt Location Register

5.21.2.9 __IO uint32_t LPC_RTC_T::RTC_AUX

RTC Auxiliary control register

5.21.2.10 __IO uint32_t LPC_RTC_T::RTC_AUXEN

RTC Auxiliary Enable register

5.21.2.11 __IO uint32_t LPC_RTC_T::TIME[RTC_TIMETYPE_LAST]

Timer field registers

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

- rtc_17xx_40xx.h

5.22 LPC_SYSCTL_T Struct Reference

LPC17XX/40XX Clock and Power register block structure.

`#include <sysctl_17xx_40xx.h>`

Public Attributes

- __IO uint32_t FLASHCFG
- uint32_t **RESERVED0** [15]
- __IO uint32_t MEMMAP
- uint32_t **RESERVED1** [15]
- SYSCTL_PLL_REGS_T PLL [SYSCTL_USB_PLL+1]
- __IO uint32_t PCON
- __IO uint32_t PCONP
- __IO uint32_t PCONP1
- uint32_t **RESERVED2** [13]
- __IO uint32_t EMCCLKSEL
- __IO uint32_t CCLKSEL
- __IO uint32_t USBCLKSEL
- __IO uint32_t CLKSRCSEL
- __IO uint32_t CANSLEEPCLR
- __IO uint32_t CANWAKEFLAGS
- uint32_t **RESERVED3** [10]
- __IO uint32_t EXTINT
- uint32_t **RESERVED4**
- __IO uint32_t EXTMODE
- __IO uint32_t EXTPOLAR
- uint32_t **RESERVED5** [12]
- __IO uint32_t RSID
- __IO uint32_t SCS
- __IO uint32_t **RESERVED7**
- __IO uint32_t PCLKSEL

- uint32_t **RESERVED9**
- __IO uint32_t PBOOST
- __IO uint32_t **SPIFICLKSEL**
- __IO uint32_t LCD_CFG
- uint32_t **RESERVED10**
- __IO uint32_t USBIntSt
- __IO uint32_t DMAREQSEL
- __IO uint32_t CLKOUTCFG
- __IO uint32_t RSTCON [2]
- uint32_t **RESERVED11** [2]
- __IO uint32_t EMCDLYCTL
- __IO uint32_t EMCCAL

5.22.1 Detailed Description

LPC17XX/40XX Clock and Power register block structure.

5.22.2 Member Data Documentation

5.22.2.1 __IO uint32_t LPC_SYSCTL_T::CANSLEEPCLR

Offset: 0x110 (R/W) CAN Sleep Clear Register

5.22.2.2 __IO uint32_t LPC_SYSCTL_T::CANWAKEFLAGS

Offset: 0x114 (R/W) CAN Wake-up Flags Register

5.22.2.3 __IO uint32_t LPC_SYSCTL_T::CCLKSEL

Offset: 0x104 (R/W) CPU Clock Selection Register

5.22.2.4 __IO uint32_t LPC_SYSCTL_T::CLKOUTCFG

Offset: 0x1C8 (R/W) Clock Output Configuration Register

5.22.2.5 __IO uint32_t LPC_SYSCTL_T::CLKSRCSEL

Offset: 0x10C (R/W) Clock Source Select Register

5.22.2.6 __IO uint32_t LPC_SYSCTL_T::DMAREQSEL

Offset: 0x1C4 (R/W) DMA Request Select Register

5.22.2.7 __IO uint32_t LPC_SYSCTL_T::EMCCAL

Offset: 0x1E0 (R/W) Calibration of programmable delays

5.22.2.8 __IO uint32_t LPC_SYSCTL_T::EMCCLKSEL

Offset: 0x100 (R/W) External Memory Controller Clock Selection Register

5.22.2.9 __IO uint32_t LPC_SYSCTL_T::EMCDLYCTL

Offset: 0x1DC (R/W) SDRAM programmable delays

5.22.2.10 __IO uint32_t LPC_SYSCTL_T::EXTINT

Offset: 0x140 (R/W) External Interrupt Flag Register

5.22.2.11 __IO uint32_t LPC_SYSCTL_T::EXTMODE

Offset: 0x148 (R/W) External Interrupt Mode Register

5.22.2.12 __IO uint32_t LPC_SYSCTL_T::EXTPOLAR

Offset: 0x14C (R/W) External Interrupt Polarity Register

5.22.2.13 __IO uint32_t LPC_SYSCTL_T::FLASHCFG

Offset: 0x000 (R/W) Flash Accelerator Configuration Register

5.22.2.14 __IO uint32_t LPC_SYSCTL_T::LCD_CFG

Offset: 0x1B8 (R/W) LCD Configuration and clocking control Register

5.22.2.15 __IO uint32_t LPC_SYSCTL_T::MEMMAP

Offset: 0x000 (R/W) Flash Accelerator Configuration Register

5.22.2.16 __IO uint32_t LPC_SYSCTL_T::PBOOST

Offset: 0x1B0 (R/W) Power Boost control register

5.22.2.17 __IO uint32_t LPC_SYSCTL_T::PCLKSEL

Offset: 0x1A8 (R/W) Peripheral Clock Selection Register

5.22.2.18 __IO uint32_t LPC_SYSCTL_T::PCON

Offset: 0x0C0 (R/W) Power Control Register

5.22.2.19 __IO uint32_t LPC_SYSCTL_T::PCONP

Offset: 0x0C4 (R/W) Power Control for Peripherals Register

5.22.2.20 __IO uint32_t LPC_SYSCTL_T::PCONP1

Offset: 0x0C8 (R/W) Power Control 1 for Peripherals Register

5.22.2.21 SYSCTL_PLL_REGS_T LPC_SYSCTL_T::PLL[SYSCTL_USB_PLL+1]

Offset: 0x080: PLL0 and PLL1

5.22.2.22 __IO uint32_t LPC_SYSCTL_T::RSID

Offset: 0x180 (R/W) Reset Source Identification Register

5.22.2.23 __IO uint32_t LPC_SYSCTL_T::RSTCON[2]

Offset: 0x1CC (R/W) RESET Control0/1 Registers

5.22.2.24 __IO uint32_t LPC_SYSCTL_T::SCS

Offset: 0x1A0 (R/W) System Controls and Status Register

5.22.2.25 __IO uint32_t LPC_SYSCTL_T::USBCLKSEL

Offset: 0x108 (R/W) USB Clock Selection Register

5.22.2.26 __IO uint32_t LPC_SYSCTL_T::USBIntSt

Offset: 0x1C0 (R/W) USB Interrupt Status Register

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

- sysctl_17xx_40xx.h

5.23 PINMUX_GRP_T Struct Reference

Array of IOCON pin definitions passed to Chip_IOCON_SetPinMuxing() must be in this format.

`#include <iocon_17xx_40xx.h>`

Public Attributes

- uint32_t **pingrp**:3
- uint32_t **pinnum**:5
- uint32_t **modefunc**:24

5.23.1 Detailed Description

Array of IOCON pin definitions passed to Chip_IOCON_SetPinMuxing() must be in this format.

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

- iocon_17xx_40xx.h

5.24 RTC_TIME_T Struct Reference

Public Attributes

- uint32_t **time** [RTC_TIMETYPE_LAST]

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

- rtc_17xx_40xx.h

5.25 SYSCTL_PLL_REGS_T Struct Reference

LPC17XX/40XX Clock and Power PLL register block structure.

`#include <sysctl_17xx_40xx.h>`

Public Attributes

- __IO uint32_t PLLCON
- __IO uint32_t PLLCFG
- __I uint32_t PLLSTAT
- __O uint32_t PLLFEED
- uint32_t **RESERVED1** [4]

5.25.1 Detailed Description

LPC17XX/40XX Clock and Power PLL register block structure.

5.25.2 Member Data Documentation

5.25.2.1 __IO uint32_t SYSCTL_PLL_REGS_T::PLLCFG

(R/W) PLL Configuration Register

5.25.2.2 __IO uint32_t SYSCTL_PLL_REGS_T::PLLCON

(R/W) PLL Control Register

5.25.2.3 __O uint32_t SYSCTL_PLL_REGS_T::PLLFEED

(/W) PLL Feed Register

5.25.2.4 __I uint32_t SYSCTL_PLL_REGS_T::PLLSTAT

(R/) PLL Status Register

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

- sysctl_17xx_40xx.h

Chapter 6

File Documentation

6.1 uip-conf.h File Reference

```
#include <stdint.h>
#include "webserver.h"
```

Project-specific configuration options

uIP has a number of configuration options that can be overridden for each project. These are kept in a project-specific uip-conf.h file and all configuration names have the prefix UIP_CONF.

- #define UIP_CONF_MAX_CONNECTIONS
- #define UIP_CONF_MAX_LISTENPORTS
- #define UIP_CONF_BUFFER_SIZE
- #define UIP_CONF_BYTE_ORDER
- #define UIP_CONF_LOGGING
- #define UIP_CONF_UDP
- #define UIP_CONF_UDP_CHECKSUMS
- #define UIP_CONF_STATISTICS
- typedef uint8_t u8_t
- typedef uint16_t u16_t
- typedef unsigned short uip_stats_t

6.1.1 Detailed Description

An example uIP configuration file

Author

Adam Dunkels adam@sics.se