Tabla de contenido

GPIO Public Functions

Functions

+void	**GPIO_SetDir** (uint8_t portNum, uint32_t bitValue, uint8_t dir) Set Direction for GPIO port.
+void	**GPIO_SetValue** (uint8_t portNum, uint32_t bitValue) Set Value for bits that have output direction on GPIO port.
+void	**GPIO_ClearValue** (uint8_t portNum, uint32_t bitValue) Clear Value for bits that have output direction on GPIO port.
+uint32_t	**GPIO_ReadValue** (uint8_t portNum) Read Current state on port pin that have input direction of GPIO.
void	**GPIO_IntCmd** (uint8_t portNum, uint32_t bitValue, uint8_t edgeState) Enable GPIO interrupt (just used for P0.0-P0.30, P2.0-P2.13).
FunctionalState	**GPIO_GetIntStatus** (uint8_t portNum, uint32_t pinNum, uint8_t edgeState) Get GPIO Interrupt Status (just used for P0.0-P0.30, P2.0-P2.13).
void	**GPIO_ClearInt** (uint8_t portNum, uint32_t bitValue) Clear GPIO interrupt (just used for P0.0-P0.30, P2.0-P2.13).
void	**FIO_SetDir** (uint8_t portNum, uint32_t bitValue, uint8_t dir) The same with **GPIO_SetDir()**.
void	**FIO_SetValue** (uint8_t portNum, uint32_t bitValue)

	The same with **GPIO_SetValue()**.
void	**FIO_ClearValue** (uint8_t portNum, uint32_t bitValue) The same with **GPIO_ClearValue()**.
uint32_t	**FIO_ReadValue** (uint8_t portNum) The same with **GPIO_ReadValue()**.
void	**FIO_SetMask** (uint8_t portNum, uint32_t bitValue, uint8_t maskValue) Set mask value for bits in FIO port.
void	**FIO_IntCmd** (uint8_t portNum, uint32_t bitValue, uint8_t edgeState) The same with **GPIO_IntCmd()**.
FunctionalState	**FIO_GetIntStatus** (uint8_t portNum, uint32_t pinNum, uint8_t edgeState) The same with **GPIO_GetIntStatus()**.
void	**FIO_ClearInt** (uint8_t portNum, uint32_t bitValue) The same with **GPIO_ClearInt()**.
void	**FIO_HalfWordSetDir** (uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue, uint8_t dir) Set direction for FIO port in halfword accessible style.
void	**FIO_HalfWordSetMask** (uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue, uint8_t maskValue) Set mask value for bits in FIO port in halfword accessible style.
void	**FIO_HalfWordSetValue** (uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue) Set bits for FIO port in halfword accessible style.
void	**FIO_HalfWordClearValue** (uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue) Clear bits for FIO port in halfword accessible style.
uint16_t	**FIO_HalfWordReadValue** (uint8_t portNum, uint8_t halfwordNum) Read Current state on port pin that have input direction of GPIO in halfword accessible style.
void	**FIO_ByteSetDir** (uint8_t portNum, uint8_t byteNum, uint8_t bitValue, uint8_t dir) Set direction for FIO port in byte accessible style.
void	**FIO_ByteSetMask** (uint8_t portNum, uint8_t byteNum, uint8_t bitValue, uint8_t maskValue) Set mask value for bits in FIO port in byte accessible style.
void	**FIO_ByteSetValue** (uint8_t portNum, uint8_t byteNum, uint8_t bitValue) Set bits for FIO port in byte accessible style.
void	**FIO_ByteClearValue** (uint8_t portNum, uint8_t byteNum, uint8_t bitValue) Clear bits for FIO port in byte accessible style.
uint8_t	**FIO_ByteReadValue** (uint8_t portNum, uint8_t byteNum) Read Current state on port pin that have input direction of GPIO in byte accessible style.

Function Documentation

void FIO_ByteClearValue (uint8_t portNum,
uint8_t byteNum,
uint8_t bitValue
)

Clear bits for FIO port in byte accessible style.

Parameters:

`[in]`	*portNum*	Port number, in range from 0 to 4
`[in]`	*byteNum*	Byte part number, should be in range from 0 to 3
`[in]`	*bitValue*	Value that contains all bits in to clear, in range from 0 to 0xFF.

Returns:
None

Note:

- For all bits that has been set as input direction, this function will not effect.
- For all remaining bits that are not activated in bitValue (value '0') will not be effected by this function.

Definition at line 708 of file lpc17xx_gpio.c.

uint8_t FIO_ByteReadValue (uint8_t portNum, uint8_t byteNum)

Read Current state on port pin that have input direction of GPIO in byte accessible style.

Parameters:

`[in]`	*portNum*	Port number, in range from 0 to 4
`[in]`	*byteNum*	Byte part number, should be in range from 0 to 3

Returns:
Current value of FIO port pin of specified byte part. Note: Return value contain state of each port pin (bit) on that FIO regardless its direction is input or output.

Definition at line 728 of file lpc17xx_gpio.c.

void FIO_ByteSetDir (uint8_t portNum, uint8_t byteNum, uint8_t bitValue, uint8_t dir)

Set direction for FIO port in byte accessible style.

Parameters:

`[in]`	*portNum*	Port number, in range from 0 to 4
`[in]`	*byteNum*	Byte part number, should be in range from 0 to 3
`[in]`	*bitValue*	Value that contains all bits in to set direction, in range from 0 to 0xFF.
`[in]`	*dir*	Direction value, should be: • 0: Input. • 1: Output.

Returns:
None

Note: All remaining bits that are not activated in bitValue (value '0') will not be effected by this function.

Definition at line 611 of file lpc17xx_gpio.c.

void FIO_ByteSetMask (uint8_t portNum, uint8_t byteNum, uint8_t bitValue, uint8_t maskValue)

Set mask value for bits in FIO port in byte accessible style.

Parameters:

[in]	*portNum*	Port number, in range from 0 to 4
[in]	*byteNum*	Byte part number, should be in range from 0 to 3
[in]	*bitValue*	Value that contains all bits in to set mask, in range from 0 to 0xFF.
[in]	*maskValue*	Mask value contains state value for each bit: • 0: not mask. • 1: mask.

Returns:
None

Note:

- All remaining bits that are not activated in bitValue (value '0') will not be effected by this function.
- After executing this function, in mask register, value '0' on each bit enables an access to the corresponding physical pin via a read or write access, while value '1' on bit (masked) that corresponding pin will not be changed with write access and if read, will not be reflected in the updated pin.

Definition at line 649 of file lpc17xx_gpio.c.

void FIO_ByteSetValue (uint8_t portNum, uint8_t byteNum, uint8_t bitValue)

Set bits for FIO port in byte accessible style.

Parameters:

[in]	*portNum*	Port number, in range from 0 to 4
[in]	*byteNum*	Byte part number, should be in range from 0 to 3
[in]	*bitValue*	Value that contains all bits in to set, in range from 0 to 0xFF.

Returns:
None

Note:

- For all bits that has been set as input direction, this function will not effect.
- For all remaining bits that are not activated in bitValue (value '0') will not be effected by this function.

Definition at line 683 of file lpc17xx_gpio.c.

void FIO_ClearInt (uint8_t portNum, uint32_t pinNum)

The same with GPIO_ClearInt().

Definition at line 376 of file lpc17xx_gpio.c.

void FIO_ClearValue (uint8_t portNum, uint32_t bitValue)

The same with GPIO_ClearValue().

Definition at line 344 of file lpc17xx_gpio.c.

FunctionalState FIO_GetIntStatus (uint8_t portNum, uint32_t pinNum, uint8_t edgeState)

The same with GPIO_GetIntStatus().

Definition at line 368 of file lpc17xx_gpio.c.

void FIO_HalfWordClearValue (uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue)

Clear bits for FIO port in halfword accessible style.

Parameters:

[in]	*portNum*	Port number, in range from 0 to 4
[in]	*halfwordNum*	HalfWord part number, should be 0 (lower) or 1(upper)
[in]	*bitValue*	Value that contains all bits in to clear, in range from 0 to 0xFFFF.

Returns:

None

Note:

- For all bits that has been set as input direction, this function will not effect.
- For all remaining bits that are not activated in bitValue (value '0') will not be effected by this function.

Definition at line 553 of file lpc17xx_gpio.c.

uint16_t FIO_HalfWordReadValue (uint8_t portNum, uint8_t halfwordNum)

Read Current state on port pin that have input direction of GPIO in halfword accessible style.

Parameters:

[in]	*portNum*	Port number, in range from 0 to 4
[in]	*halfwordNum*	HalfWord part number, should be 0 (lower) or 1(upper)

Returns:

Current value of FIO port pin of specified halfword. Note: Return value contain state of each port pin (bit) on that FIO regardless its direction is input or output.

Definition at line 578 of file lpc17xx_gpio.c.

void FIO_HalfWordSetDir (uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue, uint8_t dir)

Set direction for FIO port in halfword accessible style.

Parameters:

[in]	*portNum*	Port number, in range from 0 to 4
[in]	*halfwordNum*	HalfWord part number, should be 0 (lower) or 1(upper)
[in]	*bitValue*	Value that contains all bits in to set direction, in range from 0 to 0xFFFF.
[in]	*dir*	Direction value, should be: • 0: Input. • 1: Output.

Returns:

None

Note: All remaining bits that are not activated in bitValue (value '0') will not be effected by this function.

Definition at line 430 of file lpc17xx_gpio.c.

void FIO_HalfWordSetMask (uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue, uint8_t maskValue)

Set mask value for bits in FIO port in halfword accessible style.

Parameters:

[in]	*portNum*	Port number, in range from 0 to 4
[in]	*halfwordNum*	HalfWord part number, should be 0 (lower) or 1(upper)
[in]	*bitValue*	Value that contains all bits in to set, in range from 0 to 0xFFFF.
[in]	*maskValue*	Mask value contains state value for each bit: • 0: not mask. • 1: mask.

Returns:

None

Note:

- All remaining bits that are not activated in bitValue (value '0') will not be effected by this function.

- After executing this function, in mask register, value '0' on each bit enables an access to the corresponding physical pin via a read or write access, while value '1' on bit (masked) that corresponding pin will not be changed with write access and if read, will not be reflected in the updated pin.

Definition at line 479 of file lpc17xx_gpio.c.

void FIO_HalfWordSetValue (uint8_t portNum, uint8_t halfwordNum, uint16_t bitValue)

Set bits for FIO port in halfword accessible style.

Parameters:

[in]	*portNum*	Port number, in range from 0 to 4
[in]	*halfwordNum*	HalfWord part number, should be 0 (lower) or 1(upper)
[in]	*bitValue*	Value that contains all bits in to set, in range from 0 to 0xFFFF.

Returns:

None

Note:

- For all bits that has been set as input direction, this function will not effect.
- For all remaining bits that are not activated in bitValue (value '0') will not be effected by this function.

Definition at line 523 of file lpc17xx_gpio.c.

void FIO_IntCmd (uint8_t portNum, uint32_t bitValue, uint8_t edgeState)

The same with GPIO_IntCmd().

Definition at line 360 of file lpc17xx_gpio.c.

uint32_t FIO_ReadValue (uint8_t portNum)

The same with GPIO_ReadValue().

Definition at line 352 of file lpc17xx_gpio.c.

void FIO_SetDir (uint8_t portNum, uint32_t bitValue, uint8_t dir)

The same with GPIO_SetDir().

Definition at line 328 of file lpc17xx_gpio.c.

void FIO_SetMask (uint8_t portNum,

uint32_t **bitValue,**
uint8_t **maskValue**
)

Set mask value for bits in FIO port.

Parameters:

`[in]`	*portNum*	Port number, in range from 0 to 4
`[in]`	*bitValue*	Value that contains all bits in to set, in range from 0 to 0xFFFFFFFF.
`[in]`	*maskValue*	Mask value contains state value for each bit: • 0: not mask. • 1: mask.

Returns:
None

Note:

- All remaining bits that are not activated in bitValue (value '0') will not be effected by this function.
- After executing this function, in mask register, value '0' on each bit enables an access to the corresponding physical pin via a read or write access, while value '1' on bit (masked) that corresponding pin will not be changed with write access and if read, will not be reflected in the updated pin.

Definition at line 398 of file lpc17xx_gpio.c.

void FIO_SetValue (uint8_t **portNum,**
uint32_t **bitValue**
)

The same with GPIO_SetValue().

Definition at line 336 of file lpc17xx_gpio.c.

void GPIO_ClearInt (uint8_t **portNum,**
uint32_t **bitValue**
)

Clear GPIO interrupt (just used for P0.0-P0.30, P2.0-P2.13).

Parameters:

`[in]`	*portNum*	Port number to read value, should be: 0 or 2
`[in]`	*bitValue*	Value that contains all bits on GPIO to enable, in range from 0 to 0xFFFFFFFF.

Returns:
None

Definition at line 311 of file lpc17xx_gpio.c.

void GPIO_ClearValue (uint8_t **portNum,**
uint32_t **bitValue**
)

Clear Value for bits that have output direction on GPIO port.

Parameters:

[in]	*portNum*	Port number value, should be in range from 0 to 4
[in]	*bitValue*	Value that contains all bits on GPIO to clear, in range from 0 to 0xFFFFFFFF. example: value 0x5 to clear bit 0 and bit 1.

Returns:
None

Note:

- For all bits that has been set as input direction, this function will not effect.
- For all remaining bits that are not activated in bitValue (value '0') will not be effected by this function.

Definition at line 224 of file lpc17xx_gpio.c.

```
FunctionalState GPIO_GetIntStatus ( uint8_t   portNum,
                                    uint32_t  pinNum,
                                    uint8_t   edgeState
                                  )
```

Get GPIO Interrupt Status (just used for P0.0-P0.30, P2.0-P2.13).

Parameters:

[in]	*portNum*	Port number to read value, should be: 0 or 2
[in]	*pinNum*	Pin number, should be: 0..30(with port 0) and 0..13 (with port 2)
[in]	*edgeState*	state of edge, should be: • 0: Rising edge • 1: Falling edge

Returns:
Bool could be:

- ENABLE: Interrupt has been generated due to a rising edge on P0.0
- DISABLE: A rising edge has not been detected on P0.0

Definition at line 290 of file lpc17xx_gpio.c.

```
void GPIO_IntCmd ( uint8_t   portNum,
                   uint32_t  bitValue,
                   uint8_t   edgeState
                 )
```

Enable GPIO interrupt (just used for P0.0-P0.30, P2.0-P2.13).

Parameters:

[in]	*portNum*	Port number to read value, should be: 0 or 2
[in]	*bitValue*	Value that contains all bits on GPIO to enable, in range from 0 to 0xFFFFFFFF.
[in]	*edgeState*	state of edge, should be: • 0: Rising edge • 1: Falling edge

Returns:

None

Definition at line 262 of file lpc17xx_gpio.c.

uint32_t GPIO_ReadValue (uint8_t portNum)

Read Current state on port pin that have input direction of GPIO.

Parameters:

[in]	*portNum*	Port number to read value, in range from 0 to 4

Returns:

Current value of GPIO port.

Note: Return value contain state of each port pin (bit) on that GPIO regardless its direction is input or output.

Definition at line 241 of file lpc17xx_gpio.c.

void GPIO_SetDir (uint8_t portNum, uint32_t bitValue, uint8_t dir)

Set Direction for GPIO port.

Parameters:

[in]	*portNum*	Port Number value, should be in range from 0 to 4
[in]	*bitValue*	Value that contains all bits to set direction, in range from 0 to 0xFFFFFFFF. example: value 0x5 to set direction for bit 0 and bit 1.
[in]	*dir*	Direction value, should be: • 0: Input. • 1: Output.

Returns:

None

Note: All remaining bits that are not activated in bitValue (value '0') will not be effected by this function.

Definition at line 170 of file lpc17xx_gpio.c.

void GPIO_SetValue (uint8_t portNum, uint32_t bitValue)

Set Value for bits that have output direction on GPIO port.

Parameters:

[in]	*portNum*	Port number value, should be in range from 0 to 4
[in]	*bitValue*	Value that contains all bits on GPIO to set, in range from 0 to 0xFFFFFFFF. example: value 0x5 to set bit 0 and bit 1.

Returns:

None

Note:

- For all bits that has been set as input direction, this function will not effect.
- For all remaining bits that are not activated in bitValue (value '0') will not be effected by this function.

Definition at line **201** of file **lpc17xx_gpio.c**.

SYSTICK Public Function

Functions

+void	**SYSTICK_InternalInit** (uint32_t time) Initial System Tick with using internal CPU clock source.
void	**SYSTICK_ExternalInit** (uint32_t freq, uint32_t time) Initial System Tick with using external clock source.
+void	**SYSTICK_Cmd** (**FunctionalState** NewState) Enable/disable System Tick counter.
+void	**SYSTICK_IntCmd** (**FunctionalState** NewState) Enable/disable System Tick interrupt.
uint32_t	**SYSTICK_GetCurrentValue** (void) Get current value of System Tick counter.
void	**SYSTICK_ClearCounterFlag** (void) Clear Counter flag.

Function Documentation

void SYSTICK_ClearCounterFlag (void)

Clear Counter flag.

Parameters:
`[in]` *None*

Returns:
None

Definition at line **165** of file **lpc17xx_systick.c**.

void SYSTICK_Cmd (FunctionalState NewState)

Enable/disable System Tick counter.

Parameters:
`[in]` *NewState* System Tick counter status, should be:

- ENABLE
- DISABLE

Returns:
None

Definition at line **119** of file **lpc17xx_systick.c**.

void SYSTICK_ExternalInit (uint32_t freq ,

uint32_t time
)

Initial System Tick with using external clock source.

Parameters:

`[in]` *freq* external clock frequency(Hz)
`[in]` *time* time interval(ms)

Returns:

None

Definition at line 85 of file lpc17xx_systick.c.

uint32_t SYSTICK_GetCurrentValue (void)

Get current value of System Tick counter.

Parameters:

`[in]` *None*

Returns:

current value of System Tick counter

Definition at line 155 of file lpc17xx_systick.c.

void SYSTICK_IntCmd (FunctionalState NewState)

Enable/disable System Tick interrupt.

Parameters:

`[in]` *NewState* System Tick interrupt status, should be:

- ENABLE
- DISABLE

Returns:

None

Definition at line 138 of file lpc17xx_systick.c.

void SYSTICK_InternalInit (uint32_t time)

Initial System Tick with using internal CPU clock source.

Parameters:

`[in]` *time* time interval(ms)

Returns:

None

Definition at line 51 of file lpc17xx_systick.c.

TIM Public Functions

Functions

+void	TIM_Init (LPC_TIM_TypeDef *TIMx, TIM_MODE_OPT TimerCounterMode, void *TIM_ConfigStruct) Initial Timer/Counter device Set Clock frequency for Timer Set initial configuration for Timer.
void	TIM_DeInit (LPC_TIM_TypeDef *TIMx) Close Timer/Counter device.
void	TIM_ClearIntPending (LPC_TIM_TypeDef *TIMx, TIM_INT_TYPE IntFlag) Clear Interrupt pending.
void	TIM_ClearIntCapturePending (LPC_TIM_TypeDef *TIMx, TIM_INT_TYPE IntFlag) Clear Capture Interrupt pending.
FlagStatus	TIM_GetIntStatus (LPC_TIM_TypeDef *TIMx, TIM_INT_TYPE IntFlag) Get Interrupt Status.
FlagStatus	TIM_GetIntCaptureStatus (LPC_TIM_TypeDef *TIMx, TIM_INT_TYPE IntFlag) Get Capture Interrupt Status.
void	TIM_ConfigStructInit (TIM_MODE_OPT TimerCounterMode, void *TIM_ConfigStruct) Configuration for Timer at initial time.
+void	TIM_ConfigMatch (LPC_TIM_TypeDef *TIMx, TIM_MATCHCFG_Type *TIM_MatchConfigStruct) Configuration for Match register.
void	TIM_UpdateMatchValue (LPC_TIM_TypeDef *TIMx, uint8_t MatchChannel, uint32_t MatchValue) Update Match value.
void	TIM_SetMatchExt (LPC_TIM_TypeDef *TIMx, TIM_EXTMATCH_OPT ext_match)
void	TIM_ConfigCapture (LPC_TIM_TypeDef *TIMx, TIM_CAPTURECFG_Type *TIM_CaptureConfigStruct) Configuration for Capture register.
+void	TIM_Cmd (LPC_TIM_TypeDef *TIMx, FunctionalState NewState) Start/Stop Timer/Counter device.
uint32_t	TIM_GetCaptureValue (LPC_TIM_TypeDef *TIMx, TIM_COUNTER_INPUT_OPT CaptureChannel) Read value of capture register in timer/counter device.
void	TIM_ResetCounter (LPC_TIM_TypeDef *TIMx) Reset Timer/Counter device, Make TC and PC are synchronously reset on the next positive edge of PCLK.

Function Documentation

void TIM_ClearIntCapturePending (LPC_TIM_TypeDef * TIMx,
TIM_INT_TYPE IntFlag

)

Clear Capture Interrupt pending.

Parameters:

`[in]` *TIMx* Timer selection, should be

- LPC_TIM0: TIMER0 peripheral
  - LPC_TIM1: TIMER1 peripheral
  - LPC_TIM2: TIMER2 peripheral
  - LPC_TIM3: TIMER3 peripheral

`[in]` *IntFlag* interrupt type, should be:

- TIM_MR0_INT: Interrupt for Match channel 0
- TIM_MR1_INT: Interrupt for Match channel 1
- TIM_MR2_INT: Interrupt for Match channel 2
- TIM_MR3_INT: Interrupt for Match channel 3
- TIM_CR0_INT: Interrupt for Capture channel 0
- TIM_CR1_INT: Interrupt for Capture channel 1

Returns:

None

Definition at line **235** of file **lpc17xx_timer.c**.

void TIM_ClearIntPending (LPC_TIM_TypeDef * TIMx,
TIM_INT_TYPE IntFlag
)

Clear Interrupt pending.

Parameters:

`[in]` *TIMx* Timer selection, should be:

- LPC_TIM0: TIMER0 peripheral
  - LPC_TIM1: TIMER1 peripheral
  - LPC_TIM2: TIMER2 peripheral
  - LPC_TIM3: TIMER3 peripheral

`[in]` *IntFlag,:* interrupt type, should be:

- TIM_MR0_INT: Interrupt for Match channel 0
- TIM_MR1_INT: Interrupt for Match channel 1
- TIM_MR2_INT: Interrupt for Match channel 2
- TIM_MR3_INT: Interrupt for Match channel 3
- TIM_CR0_INT: Interrupt for Capture channel 0
- TIM_CR1_INT: Interrupt for Capture channel 1

Returns:

None

Definition at line **212** of file **lpc17xx_timer.c**.

void TIM_Cmd (LPC_TIM_TypeDef * TIMx,

FunctionalState **NewState**
)

Start/Stop Timer/Counter device.

Parameters:

[in]	*TIMx*	Pointer to **timer** device, should be: • LPC_TIM0: TIMER0 peripheral ○ LPC_TIM1: TIMER1 peripheral ○ LPC_TIM2: TIMER2 peripheral ○ LPC_TIM3: TIMER3 peripheral
[in]	*NewState*	• ENABLE : set **timer** enable • DISABLE : disable **timer**

Returns:
None

Definition at line **396** of file **lpc17xx_timer.c**.

void TIM_ConfigCapture (LPC_TIM_TypeDef * TIMx,
TIM_CAPTURECFG_Type * TIM_CaptureConfigStruct
)

Configuration for Capture register.

Parameters:

[in]	*TIMx*	Pointer to **timer** device, should be: • LPC_TIM0: TIMER0 peripheral ○ LPC_TIM1: TIMER1 peripheral ○ LPC_TIM2: TIMER2 peripheral ○ LPC_TIM3: TIMER3 peripheral ▪ CaptureChannel: set the channel to capture data ▪ RisingEdge : if SET, Capture at rising edge ▪ FallingEdge : if SET, Capture at falling edge ▪ IntOnCaption : if SET, Capture generate interrupt
[in]	*TIM_CaptureConfigStruct*	Pointer to **TIM_CAPTURECFG_Type**

Returns:
None

Definition at line **542** of file **lpc17xx_timer.c**.

void TIM_ConfigMatch (LPC_TIM_TypeDef * TIMx,
TIM_MATCHCFG_Type * TIM_MatchConfigStruct
)

Configuration for Match register.

Parameters:

[in]	*TIMx*	Pointer to **timer** device, should be:

- LPC_TIM0: TIMER0 peripheral
    - LPC_TIM1: TIMER1 peripheral
    - LPC_TIM2: TIMER2 peripheral
    - LPC_TIM3: TIMER3 peripheral

[in] *TIM_MatchConfigStruct* Pointer to **TIM_MATCHCFG_Type**

- MatchChannel : choose channel 0 or 1
- IntOnMatch : if SET, interrupt will be generated when MRxx match the value in TC
- StopOnMatch : if SET, TC and PC will be stopped whenM Rxx match the value in TC
- ResetOnMatch : if SET, Reset on MR0 when MRxx match the value in TC -ExtMatchOutputType: Select output for external match + 0: Do nothing for external output pin if match + 1: Force external output pin to low if match + 2: Force external output pin to high if match + 3: Toggle external output pin if match MatchValue: Set the value to be compared with TC value

Returns:
None

Definition at line **450** of file **lpc17xx_timer.c**.

void TIM_ConfigStructInit (TIM_MODE_OPT **TimerCounterMode ,**
void * **TIM_ConfigStruct**
)

Configuration for Timer at initial time.

Parameters:
[in] *TimerCounterMode* **timer** counter mode, should be:

- TIM_TIMER_MODE: Timer mode
- TIM_COUNTER_RISING_MODE: Counter rising mode
- TIM_COUNTER_FALLING_MODE: Counter falling mode
- TIM_COUNTER_ANY_MODE:Counter on both edges

[in] *TIM_ConfigStruct* pointer to **TIM_TIMERCFG_Type** or **TIM_COUNTERCFG_Type**

Returns:
None

Definition at line **253** of file **lpc17xx_timer.c**.

void TIM_DeInit (LPC_TIM_TypeDef * TIMx)

Close Timer/Counter device.

Parameters:
[in] *TIMx* Pointer to **timer** device, should be:

- LPC_TIM0: TIMER0 peripheral
- LPC_TIM1: TIMER1 peripheral

- LPC_TIM2: TIMER2 peripheral
- LPC_TIM3: TIMER3 peripheral

Returns:
None

Definition at line 363 of file lpc17xx_timer.c.

uint32_t TIM_GetCaptureValue (LPC_TIM_TypeDef * TIMx,
TIM_COUNTER_INPUT_OPT CaptureChannel
)

Read value of capture register in timer/counter device.

Parameters:

`[in]` *TIMx* Pointer to timer/counter device, should be:

- LPC_TIM0: TIMER0 peripheral
  - LPC_TIM1: TIMER1 peripheral
  - LPC_TIM2: TIMER2 peripheral
  - LPC_TIM3: TIMER3 peripheral

`[in]` *CaptureChannel,:* capture channel number, should be:

- TIM_COUNTER_INCAP0: CAPn.0 input pin for TIMERn
- TIM_COUNTER_INCAP1: CAPn.1 input pin for TIMERn

Returns:
Value of capture register

Definition at line 570 of file lpc17xx_timer.c.

FlagStatus TIM_GetIntCaptureStatus (LPC_TIM_TypeDef * TIMx,
TIM_INT_TYPE IntFlag
)

Get Capture Interrupt Status.

Parameters:

`[in]` *TIMx* Timer selection, should be:

- LPC_TIM0: TIMER0 peripheral
  - LPC_TIM1: TIMER1 peripheral
  - LPC_TIM2: TIMER2 peripheral
  - LPC_TIM3: TIMER3 peripheral

`[in]` *IntFlag,:* interrupt type, should be:

- TIM_MR0_INT: Interrupt for Match channel 0
- TIM_MR1_INT: Interrupt for Match channel 1
- TIM_MR2_INT: Interrupt for Match channel 2
- TIM_MR3_INT: Interrupt for Match channel 3
- TIM_CR0_INT: Interrupt for Capture channel 0

- TIM_CR1_INT: Interrupt for Capture channel 1

Returns:

FlagStatus

- SET : interrupt
- RESET : no interrupt

Definition at line 186 of file lpc17xx_timer.c.

FlagStatus TIM_GetIntStatus (LPC_TIM_TypeDef * TIMx, TIM_INT_TYPE IntFlag)

Get Interrupt Status.

Parameters:

[in] *TIMx* Timer selection, should be:

- LPC_TIM0: TIMER0 peripheral
  - LPC_TIM1: TIMER1 peripheral
  - LPC_TIM2: TIMER2 peripheral
  - LPC_TIM3: TIMER3 peripheral

[in] *IntFlag,:* interrupt type, should be:

- TIM_MR0_INT: Interrupt for Match channel 0
- TIM_MR1_INT: Interrupt for Match channel 1
- TIM_MR2_INT: Interrupt for Match channel 2
- TIM_MR3_INT: Interrupt for Match channel 3
- TIM_CR0_INT: Interrupt for Capture channel 0
- TIM_CR1_INT: Interrupt for Capture channel 1

Returns:

FlagStatus

- SET : interrupt
- RESET : no interrupt

Definition at line 156 of file lpc17xx_timer.c.

void TIM_Init (LPC_TIM_TypeDef * TIMx, TIM_MODE_OPT TimerCounterMode, void * TIM_ConfigStruct)

Initial Timer/Counter device Set Clock frequency for Timer Set initial configuration for Timer.

Parameters:

[in] *TIMx* Timer selection, should be:

- LPC_TIM0: TIMER0 peripheral
- LPC_TIM1: TIMER1 peripheral

- LPC_TIM2: TIMER2 peripheral
- LPC_TIM3: TIMER3 peripheral

[in] *TimerCounterMode* Timer counter mode, should be:

- TIM_TIMER_MODE: Timer mode
- TIM_COUNTER_RISING_MODE: Counter rising mode
- TIM_COUNTER_FALLING_MODE: Counter falling mode
- TIM_COUNTER_ANY_MODE:Counter on both edges

[in] *TIM_ConfigStruct* pointer to **TIM_TIMERCFG_Type** that contains the configuration information for the specified Timer peripheral.

Returns:
None

Definition at line **287** of file **lpc17xx_timer.c**.

void TIM_ResetCounter (LPC_TIM_TypeDef * TIMx)

Reset Timer/Counter device, Make TC and PC are synchronously reset on the next positive edge of PCLK.

Parameters:
[in] *TIMx* Pointer to **timer** device, should be:

- LPC_TIM0: TIMER0 peripheral
    - LPC_TIM1: TIMER1 peripheral
    - LPC_TIM2: TIMER2 peripheral
    - LPC_TIM3: TIMER3 peripheral

Returns:
None

Definition at line **420** of file **lpc17xx_timer.c**.

void TIM_SetMatchExt (LPC_TIM_TypeDef * TIMx, TIM_EXTMATCH_OPT ext_match)

void TIM_UpdateMatchValue (LPC_TIM_TypeDef * TIMx, uint8_t MatchChannel, uint32_t MatchValue)

Update Match value.

Parameters:
[in] *TIMx* Pointer to **timer** device, should be:

- LPC_TIM0: TIMER0 peripheral
    - LPC_TIM1: TIMER1 peripheral
    - LPC_TIM2: TIMER2 peripheral
    - LPC_TIM3: TIMER3 peripheral

[in] *MatchChannel* Match channel, should be: 0..3

[in]	*MatchValue*	updated match value

Returns:

None

Definition at line 505 of file lpc17xx_timer.c.

TIM_TIMERCFG_Type Struct Reference

Configuration structure in TIMER mode. More...

```
#include <lpc17xx_timer.h>
```

Data Fields

uint8_t	PrescaleOption
uint8_t	Reserved [3]
uint32_t	PrescaleValue

Detailed Description

Configuration structure in TIMER mode.

Definition at line 224 of file lpc17xx_timer.h.

Field Documentation

uint8_t PrescaleOption

Timer Prescale option, should be:

- TIM_PRESCALE_TICKVAL: Prescale in absolute value
- TIM_PRESCALE_USVAL: Prescale in microsecond value

Definition at line 227 of file lpc17xx_timer.h.

uint32_t PrescaleValue

Prescale value

Definition at line 232 of file lpc17xx_timer.h.

uint8_t Reserved[3]

Reserved

Definition at line 231 of file lpc17xx_timer.h.

TIM_MATCHCFG_Type Struct Reference

Match channel configuration structure. More...

`#include <lpc17xx_timer.h>`

Data Fields

uint8_t	MatchChannel
uint8_t	IntOnMatch
uint8_t	StopOnMatch
uint8_t	ResetOnMatch
uint8_t	ExtMatchOutputType
uint8_t	Reserved [3]
uint32_t	MatchValue

Detailed Description

Match channel configuration structure.

Definition at line 247 of file lpc17xx_timer.h.

Field Documentation

uint8_t ExtMatchOutputType

External Match Output type, should be:

- TIM_EXTMATCH_NOTHING: Do nothing for external output pin if match
- TIM_EXTMATCH_LOW: Force external output pin to low if match
- TIM_EXTMATCH_HIGH: Force external output pin to high if match
- TIM_EXTMATCH_TOGGLE: Toggle external output pin if match.

Definition at line 263 of file lpc17xx_timer.h.

uint8_t IntOnMatch

Interrupt On match, should be:

- ENABLE: Enable this function.
- DISABLE: Disable this function.

Definition at line 250 of file lpc17xx_timer.h.

uint8_t MatchChannel

Match channel, should be in range from 0..3

Definition at line 248 of file lpc17xx_timer.h.

uint32_t MatchValue

Reserved

Definition at line 270 of file lpc17xx_timer.h.

uint8_t Reserved[3]

Definition at line 269 of file lpc17xx_timer.h.

uint8_t ResetOnMatch

Reset On match, should be:

- ENABLE: Enable this function.
- DISABLE: Disable this function.

Definition at line 258 of file lpc17xx_timer.h.

uint8_t StopOnMatch

Stop On match, should be:

- ENABLE: Enable this function.
- DISABLE: Disable this function.

Definition at line 254 of file lpc17xx_timer.h.

UART Public Functions

Functions

+void	**UART_Init** (**LPC_UART_TypeDef** *UARTx, **UART_CFG_Type** *UART_ConfigStruct) Initializes the UARTx peripheral according to the specified parameters in the UART_ConfigStruct.
void	**UART_DeInit** (**LPC_UART_TypeDef** *UARTx) De-initializes the UARTx peripheral registers to their default reset values.
+void	**UART_ConfigStructInit** (**UART_CFG_Type** *UART_InitStruct) Fills each UART_InitStruct member with its default value: • 9600 bps • 8-bit data • 1 Stopbit • None Parity.
void	**UART_SendByte** (**LPC_UART_TypeDef** *UARTx, uint8_t Data) Transmit a single data through UART peripheral.
uint8_t	**UART_ReceiveByte** (**LPC_UART_TypeDef** *UARTx) Receive a single data from UART peripheral.
+uint32_t	**UART_Send** (**LPC_UART_TypeDef** *UARTx, uint8_t *txbuf, uint32_t buflen, **TRANSFER_BLOCK_Type** flag) Send a block of data via UART peripheral.
+uint32_t	**UART_Receive** (**LPC_UART_TypeDef** *UARTx, uint8_t *rxbuf, uint32_t buflen, **TRANSFER_BLOCK_Type** flag) Receive a block of data via UART peripheral.
+void	**UART_FIFOConfig** (**LPC_UART_TypeDef** *UARTx, **UART_FIFO_CFG_Type** *FIFOCfg) Configure FIFO function on selected UART peripheral.
+void	**UART_FIFOConfigStructInit** (**UART_FIFO_CFG_Type** *UART_FIFOInitStruct) Fills each UART_FIFOInitStruct member with its default value: • FIFO_DMAMode = DISABLE • FIFO_Level = UART_FIFO_TRGLEV0 • FIFO_ResetRxBuf = ENABLE • FIFO_ResetTxBuf = ENABLE • FIFO_State = ENABLE.
uint32_t	**UART_GetIntId** (**LPC_UART_TypeDef** *UARTx) Get Interrupt Identification value.
uint8_t	**UART_GetLineStatus** (**LPC_UART_TypeDef** *UARTx)

	Get current value of Line Status register in UART peripheral.
void	**UART_IntConfig** (**LPC_UART_TypeDef** *UARTx, **UART_INT_Type** UARTIntCfg, **FunctionalState** NewState) Enable or disable specified UART interrupt.
+void	**UART_TxCmd** (**LPC_UART_TypeDef** *UARTx, **FunctionalState** NewState) Enable/Disable transmission on UART TxD pin.
FlagStatus	**UART_CheckBusy** (**LPC_UART_TypeDef** *UARTx) Check whether if UART is busy or not.
void	**UART_ForceBreak** (**LPC_UART_TypeDef** *UARTx) Force BREAK character on UART line, output pin UARTx TXD is forced to logic 0.
void	**UART_ABClearIntPending** (**LPC_UART_TypeDef** *UARTx, **UART_ABEO_Type** ABIntType) Clear Autobaud Interrupt Pending.
void	**UART_ABCmd** (**LPC_UART_TypeDef** *UARTx, **UART_AB_CFG_Type** *ABConfigStruct, **FunctionalState** NewState) Start/Stop Auto Baudrate activity.
void	**UART_FullModemForcePinState** (**LPC_UART1_TypeDef** *UARTx, **UART_MODEM_PIN_Type** Pin, **UART1_SignalState** NewState) Force pin DTR/RTS corresponding to given state (Full modem mode).
void	**UART_FullModemConfigMode** (**LPC_UART1_TypeDef** *UARTx, **UART_MODEM_MODE_Type** Mode, **FunctionalState** NewState) Configure Full Modem mode for UART peripheral.
uint8_t	**UART_FullModemGetStatus** (**LPC_UART1_TypeDef** *UARTx) Get current status of modem status register.
void	**UART_RS485Config** (**LPC_UART1_TypeDef** *UARTx, **UART1_RS485_CTRLCFG_Type** *RS485ConfigStruct) Configure UART peripheral in RS485 mode according to the specified parameters in the RS485ConfigStruct.
void	**UART_RS485ReceiverCmd** (**LPC_UART1_TypeDef** *UARTx, **FunctionalState** NewState) Enable/Disable receiver in RS485 module in UART1.
void	**UART_RS485SendSlvAddr** (**LPC_UART1_TypeDef** *UARTx, uint8_t SlvAddr) Send Slave address frames on RS485 bus.
uint32_t	**UART_RS485SendData** (**LPC_UART1_TypeDef** *UARTx, uint8_t *pData, uint32_t size) Send Data frames on RS485 bus.
void	**UART_IrDAInvtInputCmd** (**LPC_UART_TypeDef** *UARTx, **FunctionalState** NewState) Enable or disable inverting serial input function of IrDA on UART peripheral.
void	**UART_IrDACmd** (**LPC_UART_TypeDef** *UARTx, **FunctionalState** NewState) Enable or disable IrDA function on UART peripheral.
void	**UART_IrDAPulseDivConfig** (**LPC_UART_TypeDef** *UARTx, **UART_IrDA_PULSE_Type** PulseDiv)

	Configure Pulse divider for IrDA function on UART peripheral.
uint32_t	**UART_RS485Send** (**LPC_UART1_TypeDef** *UARTx, uint8_t *pDatFrm, uint32_t size, uint8_t ParityStick) Send data on RS485 bus with specified parity stick value (9-bit mode).

Function Documentation

void UART_ABClearIntPending (LPC_UART_TypeDef * UARTx,
UART_ABEO_Type ABIntType)

Clear Autobaud Interrupt Pending.

Parameters:

[in] *UARTx* UART peripheral selected, should be

- LPC_UART0: UART0 peripheral
  - LPC_UART1: UART1 peripheral
  - LPC_UART2: UART2 peripheral
  - LPC_UART3: UART3 peripheral

[in] *ABIntType* type of auto-baud interrupt, should be:

- UART_AUTOBAUD_INTSTAT_ABEO: End of Auto-baud interrupt
- UART_AUTOBAUD_INTSTAT_ABTO: Auto-baud time out interrupt

Returns:
none

Definition at line **967** of file **lpc17xx_uart.c**.

void UART_ABCmd (LPC_UART_TypeDef * UARTx,
UART_AB_CFG_Type * ABConfigStruct,
FunctionalState NewState)

Start/Stop Auto Baudrate activity.

Parameters:

[in] *UARTx* UART peripheral selected, should be

- LPC_UART0: UART0 peripheral
  - LPC_UART1: UART1 peripheral
  - LPC_UART2: UART2 peripheral
  - LPC_UART3: UART3 peripheral

[in] *ABConfigStruct* A pointer to **UART_AB_CFG_Type** structure that contains specified information about UART auto baudrate configuration

[in] *NewState* New State of Auto baudrate activity, should be:

- ENABLE: Start this activity
- DISABLE: Stop this activity Note: Auto-baudrate mode enable bit will be cleared

once this mode completed.

Returns:
none

Definition at line 899 of file lpc17xx_uart.c.

FlagStatus UART_CheckBusy (LPC_UART_TypeDef * UARTx)

Check whether if UART is busy or not.

Parameters:
[in] *UARTx* UART peripheral selected, should be:

- LPC_UART0: UART0 peripheral
  - LPC_UART1: UART1 peripheral
  - LPC_UART2: UART2 peripheral
  - LPC_UART3: UART3 peripheral

Returns:
RESET if UART is not busy, otherwise return SET.

Definition at line 788 of file lpc17xx_uart.c.

void UART_ConfigStructInit (UART_CFG_Type * UART_InitStruct)

Fills each UART_InitStruct member with its default value:

- 9600 bps
- 8-bit data
- 1 Stopbit
- None Parity.

Parameters:
[in] *UART_InitStruct* Pointer to a UART_CFG_Type structure which will be initialized.

Returns:
None

Definition at line 442 of file lpc17xx_uart.c.

void UART_DeInit (LPC_UART_TypeDef * UARTx)

De-initializes the UARTx peripheral registers to their default reset values.

Parameters:
[in] *UARTx* UART peripheral selected, should be:

- LPC_UART0: UART0 peripheral
  - LPC_UART1: UART1 peripheral
  - LPC_UART2: UART2 peripheral
  - LPC_UART3: UART3 peripheral

Returns:
None

Definition at line 392 of file lpc17xx_uart.c.

void UART_FIFOConfig (LPC_UART_TypeDef * UARTx,
UART_FIFO_CFG_Type * FIFOCfg
)

Configure FIFO function on selected UART peripheral.

Parameters:

[in] *UARTx* UART peripheral selected, should be:

- LPC_UART0: UART0 peripheral
  - LPC_UART1: UART1 peripheral
  - LPC_UART2: UART2 peripheral
  - LPC_UART3: UART3 peripheral

[in] *FIFOCfg* Pointer to a UART_FIFO_CFG_Type Structure that contains specified information about FIFO configuration

Returns:

none

Definition at line 809 of file lpc17xx_uart.c.

void UART_FIFOConfigStructInit (UART_FIFO_CFG_Type * UART_FIFOInitStruct)

Fills each UART_FIFOInitStruct member with its default value:

- FIFO_DMAMode = DISABLE
- FIFO_Level = UART_FIFO_TRGLEV0
- FIFO_ResetRxBuf = ENABLE
- FIFO_ResetTxBuf = ENABLE
- FIFO_State = ENABLE.

Parameters:

[in] *UART_FIFOInitStruct* Pointer to a UART_FIFO_CFG_Type structure which will be initialized.

Returns:

None

Definition at line 873 of file lpc17xx_uart.c.

void UART_ForceBreak (LPC_UART_TypeDef * UARTx)

Force BREAK character on UART line, output pin UARTx TXD is forced to logic 0.

Parameters:

[in] *UARTx* UART peripheral selected, should be:

- LPC_UART0: UART0 peripheral
  - LPC_UART1: UART1 peripheral
  - LPC_UART2: UART2 peripheral
  - LPC_UART3: UART3 peripheral

Returns:

None

Definition at line 632 of file lpc17xx_uart.c.

void UART_FullModemConfigMode (LPC_UART1_TypeDef * UARTx, UART_MODEM_MODE_Type Mode, FunctionalState NewState)

Configure Full Modem mode for UART peripheral.

Parameters:

[in]	*UARTx*	LPC_UART1 (only)
[in]	*Mode*	Full Modem mode, should be: • UART1_MODEM_MODE_LOOPBACK: Loop back mode. • UART1_MODEM_MODE_AUTO_RTS: Auto-RTS mode. • UART1_MODEM_MODE_AUTO_CTS: Auto-CTS mode.
[in]	*NewState*	New State of this mode, should be: • ENABLE: Enable this mode. • DISABLE: Disable this mode.

Returns:
none

Definition at line 1158 of file lpc17xx_uart.c.

void UART_FullModemForcePinState (LPC_UART1_TypeDef * UARTx, UART_MODEM_PIN_Type Pin, UART1_SignalState NewState)

Force pin DTR/RTS corresponding to given state (Full modem mode).

Parameters:

[in]	*UARTx*	LPC_UART1 (only)
[in]	*Pin*	Pin that NewState will be applied to, should be: • UART1_MODEM_PIN_DTR: DTR pin. • UART1_MODEM_PIN_RTS: RTS pin.
[in]	*NewState*	New State of DTR/RTS pin, should be: • INACTIVE: Force the pin to inactive signal. • ACTIVE: Force the pin to active signal.

Returns:
none

Definition at line 1118 of file lpc17xx_uart.c.

uint8_t UART_FullModemGetStatus (LPC_UART1_TypeDef * UARTx)

Get current status of modem status register.

Parameters:

`[in]` *UARTx* LPC_UART1 (only)

Returns:

Current value of modem status register Note: The return value of this function must be ANDed with each member UART_MODEM_STAT_type enumeration to determine current flag status corresponding to each modem flag status. Because some flags in modem status register will be cleared after reading, the next reading modem register could not be correct. So this function used to read modem status register in one time only, then the return value used to check all flags.

Definition at line **1204** of file **lpc17xx_uart.c**.

uint32_t UART_GetIntId (LPC_UART_TypeDef * UARTx)

Get Interrupt Identification value.

Parameters:

`[in]` *UARTx* UART peripheral selected, should be:

- LPC_UART0: UART0 peripheral
  - LPC_UART1: UART1 peripheral
  - LPC_UART2: UART2 peripheral
  - LPC_UART3: UART3 peripheral

Returns:

Current value of UART UIIR register in UART peripheral.

Definition at line **773** of file **lpc17xx_uart.c**.

uint8_t UART_GetLineStatus (LPC_UART_TypeDef * UARTx)

Get current value of Line Status register in UART peripheral.

Parameters:

`[in]` *UARTx* UART peripheral selected, should be:

- LPC_UART0: UART0 peripheral
  - LPC_UART1: UART1 peripheral
  - LPC_UART2: UART2 peripheral
  - LPC_UART3: UART3 peripheral

Returns:

Current value of Line Status register in UART peripheral. Note: The return value of this function must be ANDed with each member in UART_LS_Type enumeration to determine current flag status corresponding to each Line status type. Because some flags in Line Status register will be cleared after reading, the next reading Line Status register could not be correct. So this function used to read Line status register in one time only, then the return value used to check all flags.

Definition at line **750** of file **lpc17xx_uart.c**.

void UART_Init (LPC_UART_TypeDef * UARTx,
UART_CFG_Type * UART_ConfigStruct
)

Initializes the UARTx peripheral according to the specified parameters in the UART_ConfigStruct.

Parameters:

`[in]` *UARTx* UART peripheral selected, should be:

- LPC_UART0: UART0 peripheral
  - LPC_UART1: UART1 peripheral
  - LPC_UART2: UART2 peripheral
  - LPC_UART3: UART3 peripheral

`[in]` *UART_ConfigStruct* Pointer to a **UART_CFG_Type** structure that contains the configuration information for the specified UART peripheral.

Returns:

None

Definition at line **186** of file **lpc17xx_uart.c**.

void UART_IntConfig (LPC_UART_TypeDef * UARTx, UART_INT_Type UARTIntCfg, FunctionalState NewState)

Enable or disable specified UART interrupt.

Parameters:

`[in]` *UARTx* UART peripheral selected, should be

- LPC_UART0: UART0 peripheral
  - LPC_UART1: UART1 peripheral
  - LPC_UART2: UART2 peripheral
  - LPC_UART3: UART3 peripheral

`[in]` *UARTIntCfg* Specifies the interrupt flag, should be one of the following:

- UART_INTCFG_RBR : RBR Interrupt enable
- UART_INTCFG_THRE : THR Interrupt enable
- UART_INTCFG_RLS : RX line status interrupt enable
- UART1_INTCFG_MS : Modem status interrupt enable (UART1 only)
- UART1_INTCFG_CTS : CTS1 signal transition interrupt enable (UART1 only)
- UART_INTCFG_ABEO : Enables the end of auto-baud interrupt
- UART_INTCFG_ABTO : Enables the auto-baud time-out interrupt

`[in]` *NewState* New state of specified UART interrupt type, should be:

- ENALBE: Enable this UART interrupt type.
- DISALBE: Disable this UART interrupt type.

Returns:

None

Definition at line **669** of file **lpc17xx_uart.c**.

void UART_IrDACmd (LPC_UART_TypeDef * UARTx, FunctionalState NewState)

Enable or disable IrDA function on UART peripheral.

Parameters:

`[in]`	*UARTx*	UART peripheral selected, should be LPC_UART3 (only)
`[in]`	*NewState*	New state of IrDA function, should be:

- ENABLE: Enable this function.
- DISABLE: Disable this function.

Returns:
none

Definition at line **1056** of file **lpc17xx_uart.c**.

void UART_IrDAInvtInputCmd (LPC_UART_TypeDef * UARTx,
FunctionalState NewState
)

Enable or disable inverting serial input function of IrDA on UART peripheral.

Parameters:

`[in]`	*UARTx*	UART peripheral selected, should be LPC_UART3 (only)
`[in]`	*NewState*	New state of inverting serial input, should be:

- ENABLE: Enable this function.
- DISABLE: Disable this function.

Returns:
none

Definition at line **1032** of file **lpc17xx_uart.c**.

void UART_IrDAPulseDivConfig (LPC_UART_TypeDef * UARTx,
UART_IrDA_PULSE_Type PulseDiv
)

Configure Pulse divider for IrDA function on UART peripheral.

Parameters:

`[in]`	*UARTx*	UART peripheral selected, should be LPC_UART3 (only)
`[in]`	*PulseDiv*	Pulse Divider value from Peripheral clock, should be one of the following:

- UART_IrDA_PULSEDIV2 : Pulse width = 2 * Tpclk
- UART_IrDA_PULSEDIV4 : Pulse width = 4 * Tpclk
- UART_IrDA_PULSEDIV8 : Pulse width = 8 * Tpclk
- UART_IrDA_PULSEDIV16 : Pulse width = 16 * Tpclk
- UART_IrDA_PULSEDIV32 : Pulse width = 32 * Tpclk
- UART_IrDA_PULSEDIV64 : Pulse width = 64 * Tpclk
- UART_IrDA_PULSEDIV128 : Pulse width = 128 * Tpclk
- UART_IrDA_PULSEDIV256 : Pulse width = 256 * Tpclk

Returns:
none

Definition at line 1088 of file lpc17xx_uart.c.

uint32_t UART_Receive (LPC_UART_TypeDef * UARTx, uint8_t * rxbuf, uint32_t buflen, TRANSFER_BLOCK_Type flag)

Receive a block of data via UART peripheral.

Parameters:

[in]	*UARTx*	Selected UART peripheral used to send data, should be: • LPC_UART0: UART0 peripheral ○ LPC_UART1: UART1 peripheral ○ LPC_UART2: UART2 peripheral ○ LPC_UART3: UART3 peripheral
[out]	*rxbuf*	Pointer to Received buffer
[in]	*buflen*	Length of Received buffer
[in]	*flag*	Flag mode, should be NONE_BLOCKING or BLOCKING

Returns:

Number of bytes received

Note: when using UART in BLOCKING mode, a time-out condition is used via defined symbol UART_BLOCKING_TIMEOUT.

Definition at line 581 of file lpc17xx_uart.c.

uint8_t UART_ReceiveByte (LPC_UART_TypeDef * UARTx)

Receive a single data from UART peripheral.

Parameters:

[in]	*UARTx*	UART peripheral selected, should be: • LPC_UART0: UART0 peripheral ○ LPC_UART1: UART1 peripheral ○ LPC_UART2: UART2 peripheral ○ LPC_UART3: UART3 peripheral

Returns:

Data received

Definition at line 486 of file lpc17xx_uart.c.

void UART_RS485Config (LPC_UART1_TypeDef * UARTx, UART1_RS485_CTRLCFG_Type * RS485ConfigStruct)

Configure UART peripheral in RS485 mode according to the specified parameters in the RS485ConfigStruct.

Parameters:

[in]	*UARTx*	LPC_UART1 (only)

`[in]`	*RS485ConfigStruct*	Pointer to a **UART1_RS485_CTRLCFG_Type** structure that contains the configuration information for specified UART in RS485 mode.

Returns:
None

Definition at line **1222** of file **lpc17xx_uart.c**.

void UART_RS485ReceiverCmd (LPC_UART1_TypeDef * UARTx, FunctionalState NewState)

Enable/Disable receiver in RS485 module in UART1.

Parameters:

`[in]`	*UARTx*	LPC_UART1 (only)
`[in]`	*NewState*	New State of command, should be: • ENABLE: Enable this function. • DISABLE: Disable this function.

Returns:
None

Definition at line **1294** of file **lpc17xx_uart.c**.

uint32_t UART_RS485Send (LPC_UART1_TypeDef * UARTx, uint8_t * pDatFrm, uint32_t size, uint8_t ParityStick)

Send data on RS485 bus with specified parity stick value (9-bit mode).

Parameters:

`[in]`	*UARTx*	LPC_UART1 (only)
`[in]`	*pDatFrm*	Pointer to data frame.
`[in]`	*size*	Size of data.
`[in]`	*ParityStick*	Parity Stick value, should be 0 or 1.

Returns:
None

Definition at line **1311** of file **lpc17xx_uart.c**.

uint32_t UART_RS485SendData (LPC_UART1_TypeDef * UARTx, uint8_t * pData, uint32_t size)

Send Data frames on RS485 bus.

Parameters:

`[in]`	*UARTx*	LPC_UART1 (only)

[in]	*pData*	Pointer to data to be sent.
[in]	*size*	Size of data frame to be sent.

Returns:
None

Definition at line 1349 of file lpc17xx_uart.c.

void UART_RS485SendSlvAddr (LPC_UART1_TypeDef * UARTx,
uint8_t SlvAddr
)

Send Slave address frames on RS485 bus.

Parameters:

[in]	*UARTx*	LPC_UART1 (only)
[in]	*SlvAddr*	Slave Address.

Returns:
None

Definition at line 1337 of file lpc17xx_uart.c.

uint32_t UART_Send (LPC_UART_TypeDef * UARTx,
uint8_t * txbuf,
uint32_t buflen,
TRANSFER_BLOCK_Type flag
)

Send a block of data via UART peripheral.

Parameters:
[in] *UARTx* Selected UART peripheral used to send data, should be:

- LPC_UART0: UART0 peripheral
  - LPC_UART1: UART1 peripheral
  - LPC_UART2: UART2 peripheral
  - LPC_UART3: UART3 peripheral

[in]	*txbuf*	Pointer to Transmit buffer
[in]	*buflen*	Length of Transmit buffer
[in]	*flag*	Flag used in UART transfer, should be NONE_BLOCKING or BLOCKING

Returns:
Number of bytes sent.
Note: when using UART in BLOCKING mode, a time-out condition is used via defined symbol UART_BLOCKING_TIMEOUT.

Definition at line 516 of file lpc17xx_uart.c.

void UART_SendByte (LPC_UART_TypeDef * UARTx,
uint8_t Data
)

Transmit a single data through UART peripheral.

Parameters:

`[in]` *UARTx* UART peripheral selected, should be:

- LPC_UART0: UART0 peripheral
    - LPC_UART1: UART1 peripheral
    - LPC_UART2: UART2 peripheral
    - LPC_UART3: UART3 peripheral

`[in]` *Data* Data to transmit (must be 8-bit long)

Returns:

None

Definition at line 461 of file lpc17xx_uart.c.

void UART_TxCmd (LPC_UART_TypeDef * UARTx,
FunctionalState NewState
)

Enable/Disable transmission on UART TxD pin.

Parameters:

`[in]` *UARTx* UART peripheral selected, should be:

- LPC_UART0: UART0 peripheral
    - LPC_UART1: UART1 peripheral
    - LPC_UART2: UART2 peripheral
    - LPC_UART3: UART3 peripheral

`[in]` *NewState* New State of Tx transmission function, should be:

- ENABLE: Enable this function
- DISABLE: Disable this function

Returns:

none

Definition at line 990 of file lpc17xx_uart.c.

UART_CFG_Type Struct Reference

UART Configuration Structure definition. More...

`#include <lpc17xx_uart.h>`

Data Fields

uint32_t	Baud_rate
UART_PARITY_Type	Parity
UART_DATABIT_Type	Databits
UART_STOPBIT_Type	Stopbits

Detailed Description

UART Configuration Structure definition.

Definition at line 491 of file lpc17xx_uart.h.

Field Documentation

uint32_t Baud_rate

UART baud rate

Definition at line 492 of file lpc17xx_uart.h.

UART_DATABIT_Type Databits

Number of data bits, should be:

- UART_DATABIT_5: UART 5 bit data mode
- UART_DATABIT_6: UART 6 bit data mode
- UART_DATABIT_7: UART 7 bit data mode
- UART_DATABIT_8: UART 8 bit data mode

Definition at line 500 of file lpc17xx_uart.h.

UART_PARITY_Type Parity

Parity selection, should be:

- UART_PARITY_NONE: No parity
- UART_PARITY_ODD: Odd parity
- UART_PARITY_EVEN: Even parity
- UART_PARITY_SP_1: Forced "1" stick parity
- UART_PARITY_SP_0: Forced "0" stick parity

Definition at line 493 of file lpc17xx_uart.h.

UART_STOPBIT_Type Stopbits

Number of stop bits, should be:

- UART_STOPBIT_1: UART 1 Stop Bits Select
- UART_STOPBIT_2: UART 2 Stop Bits Select

Definition at line 506 of file lpc17xx_uart.h.

UART_FIFO_CFG_Type Struct Reference

UART FIFO Configuration Structure definition. More...

```
#include <lpc17xx_uart.h>
```

Data Fields

FunctionalState	FIFO_ResetRxBuf
FunctionalState	FIFO_ResetTxBuf
FunctionalState	FIFO_DMAMode
UART_FITO_LEVEL_Type	FIFO_Level

Detailed Description

UART FIFO Configuration Structure definition.

Definition at line **516** of file **lpc17xx_uart.h**.

Field Documentation

FunctionalState FIFO_DMAMode

DMA mode, should be:

- ENABLE: Enable DMA mode in UART
- DISABLE: Disable DMA mode in UART

Definition at line **525** of file **lpc17xx_uart.h**.

UART_FITO_LEVEL_Type FIFO_Level

Rx FIFO trigger level, should be:

- UART_FIFO_TRGLEV0: UART FIFO trigger level 0: 1 character
- UART_FIFO_TRGLEV1: UART FIFO trigger level 1: 4 character
- UART_FIFO_TRGLEV2: UART FIFO trigger level 2: 8 character
- UART_FIFO_TRGLEV3: UART FIFO trigger level 3: 14 character

Definition at line **529** of file **lpc17xx_uart.h**.

FunctionalState FIFO_ResetRxBuf

Reset Rx FIFO command state , should be:

- ENABLE: Reset Rx FIFO in UART
- DISABLE: Do not reset Rx FIFO in UART

Definition at line **517** of file **lpc17xx_uart.h**.

FunctionalState FIFO_ResetTxBuf

Reset Tx FIFO command state , should be:

- ENABLE: Reset Tx FIFO in UART
- DISABLE: Do not reset Tx FIFO in UART

Definition at line **521** of file **lpc17xx_uart.h**.

PINSEL Public Functions

Functions

+void	**PINSEL_ConfigPin** (**PINSEL_CFG_Type** *PinCfg) Configure Pin corresponding to specified parameters passed in the PinCfg.
void	**PINSEL_ConfigTraceFunc** (**FunctionalState** NewState) Configure trace function.
void	**PINSEL_SetI2C0Pins** (uint8_t i2cPinMode, **FunctionalState** filterSlewRateEnable) Setup I2C0 pins.

Function Documentation

void PINSEL_ConfigPin (PINSEL_CFG_Type * PinCfg)

Configure Pin corresponding to specified parameters passed in the PinCfg.

Parameters:

[in] *PinCfg* Pointer to a **PINSEL_CFG_Type** structure that contains the configuration information for the specified pin.

Returns:

None

Definition at line **290** of file **lpc17xx_pinsel.c**.

void PINSEL_ConfigTraceFunc (FunctionalState NewState)

Configure trace function.

Parameters:

[in] *NewState* State of the Trace function configuration, should be one of the following:

- ENABLE : Enable Trace Function
- DISABLE : Disable Trace Function

Returns:

None

Definition at line **245** of file **lpc17xx_pinsel.c**.

void PINSEL_SetI2C0Pins (uint8_t i2cPinMode, FunctionalState filterSlewRateEnable)

Setup I2C0 pins.

Parameters:

[in] *i2cPinMode* I2C pin mode, should be one of the following:

- PINSEL_I2C_Normal_Mode : The standard drive mode
- PINSEL_I2C_Fast_Mode : Fast Mode Plus drive mode

`[in]` *filterSlewRateEnable* should be:

- ENABLE: Enable filter and slew rate.
- DISABLE: Disable filter and slew rate.

Returns:
None

Definition at line **267** of file **lpc17xx_pinsel.c**.

PINSEL_CFG_Type Struct Reference

Pin configuration structure. More...

`#include <lpc17xx_pinsel.h>`

Data Fields

uint8_t	**Portnum**
uint8_t	**Pinnum**
uint8_t	**Funcnum**
uint8_t	**Pinmode**
uint8_t	**OpenDrain**

Detailed Description

Pin configuration structure.

Definition at line **143** of file **lpc17xx_pinsel.h**.

Field Documentation

uint8_t Funcnum

Function Number, should be PINSEL_FUNC_x, where x should be in range from 0 to 3

Definition at line **149** of file **lpc17xx_pinsel.h**.

uint8_t OpenDrain

OpenDrain mode, should be:

- PINSEL_PINMODE_NORMAL: Pin is in the normal (not open drain) mode
- PINSEL_PINMODE_OPENDRAIN: Pin is in the open drain mode

Definition at line **155** of file **lpc17xx_pinsel.h**.

uint8_t Pinmode

Pin Mode, should be:

- PINSEL_PINMODE_PULLUP: Internal pull-up resistor
- PINSEL_PINMODE_TRISTATE: Tri-state
- PINSEL_PINMODE_PULLDOWN: Internal pull-down resistor

Definition at line **151** of file **lpc17xx_pinsel.h**.

uint8_t Pinnum

Pin Number, should be PINSEL_PIN_x, where x should be in range from 0 to 31

Definition at line **147** of file **lpc17xx_pinsel.h**.

uint8_t Portnum

Port Number, should be PINSEL_PORT_x, where x should be in range from 0 to 4

Definition at line **145** of file **lpc17xx_pinsel.h**.

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

- C:/nxpdrv/LPC1700CMSIS/Drivers/include/**lpc17xx_pinsel.h**

NVIC

```
NVIC_EnableIRQ(TIMER0_IRQn);
```